Learning Language from Perceptual Context

David L. Chen
Department of Computer Science
University of Texas at Austin
Austin, TX 78712
dlcc@cs.utexas.edu

Doctoral Dissertation Proposal

Supervising Professor: Raymond J. Mooney

Abstract

Most current natural language processing (NLP) systems are built using statistical learning algorithms trained on large annotated corpora which can be expensive and time-consuming to collect. In contrast, humans can learn language through exposure to linguistic input in the context of a rich, relevant, perceptual environment. If a machine learning system can acquire language in a similar manner without explicit human supervision, then it can leverage the large amount of available text that refers to observed world states (e.g. sportscasts, instruction manuals, weather forecasts, etc.) Thus, my research focuses on how to build systems that use both text and the perceptual context in which it is used in order to learn a language.

I will first present a system we completed that can describe events in RoboCup 2D simulation games by learning only from sample language commentaries paired with traces of simulated activities without any language-specific prior knowledge. By applying an EM-like algorithm, the system was able to simultaneously learn a grounded language model as well as align the ambiguous training data. Human evaluations of the generated commentaries indicate they are of reasonable quality and in some cases even on par with those produced by humans.

For future work, I am proposing to solve the more complex task of learning how to give and receive navigation instructions in a virtual environment. In this setting, each instruction corresponds to a navigation plan that is not directly observable. Since an exponential number of plans can all lead to the same observed actions, we have to learn from compact representations of all valid plans rather than enumerating all possible meanings as we did in the sportscasting task. Initially, the system will passively observe a human giving instruction to another human, and try to learn the correspondences between the instructions and the intended plan. After the system has a decent understanding of the language, it can then participate in the interactions to learn more directly by playing either the role of the instructor or the follower.

Contents

1 Introduction

Learning the semantics of language is one of the ultimate goals of NLP. By connecting words and phrases to objects and events in the world, the semantics of language is grounded in perceptual experience (Harnad, 1990). Although there has been some interesting computational research on such "grounded language learning" (Roy, 2002; Bailey, Feldman, Narayanan, & Lakoff, 1997; Barnard, Duygulu, Forsyth, de Freitas, Blei, & Jordan, 2003; Yu & Ballard, 2004; Gold & Scassellati, 2007), most of the focus has been on dealing with raw perceptual data, and the complexity of the language involved has been very modest. Many of the systems aim to learn meanings of words rather than interpreting entire sentences. In contrast, the goal of my research is to learn to map words and phrases into logical components that can be composed together to form complete meanings. To make the problem tractable, we study the problem in simulated environments that retain many of the important properties of a dynamic world with multiple agents and actions while avoiding many of the complexities of robotics and computer vision.

While there is existing work that learns how to translate natural language to and from a formal, logical language (Mooney, 2007; Zettlemoyer & Collins, 2007; Lu, Ng, Lee, & Zettlemoyer, 2008), they require supervised corpora where each sentence is manually translated into the formal language. Constructing such parallel corpora can be difficult and time-consuming thus limiting the effectiveness of these approaches. In contrast, humans can acquire language through simultaneous exposure to linguistic inputs and perceptions. By repeatedly hearing the same word and perceiving the same object or event in the world, a correspondence between the two can be established. Thus, we aim to build a machine learning system that can acquire language in a similar manner without explicit human supervision.

As a first step, we have developed a system that learns how to sportscast a RoboCup simulation game. The system observes how humans commentate the games and try to align each textual comment to one of the events that has just occurred. While performing the alignment, the system also tries to learn a translation model between the natural language and the representation of the events. Using an EM-like approach, the system alternates between aligning the data and building the translation model. The learned translation model allows the system to commentate on any events that occur in the game. To produce an actual sportscast, we also developed an algorithm for learning which events to describe at any given time. Evaluation of the overall system was performed by human judges recruited over the internet. Our system produced reasonable sportscasts in most cases and sometimes even on par with those produced by humans.

To expand our system to deal with more complicated domains, we will next try to solve the problem of learning navigation instructions in a virtual environment. The goal is to be able to both follow a set of instructions to reach an intended destination and to give instructions to guide a human follower to a particular location. Unlike the sportscasting task where we could directly observe the events that the comments refer to, we cannot directly observe the navigation plans that are referred to by these instructions. Instead, we must infer the possible plans from our knowledge about the environment. This results in an exponential number of possible plans that can be mapped to each instruction compared to only a handful of events that can be mapped to each comment. This explosion of ambiguity dictates that we must deal with compact representations of the possible plans rather than enumerate them as we did before for the sportscasting task. Another interesting aspect of the navigation task is that it allows for interactive learning by the system. Instead of just passively observing data, the system can participate in the process either as the instructor or the follower once it has a decent understanding of the language. Feedback from its human partner lets the system evaluate the quality of its model and improve on its problem areas.

2 Background and Related Work

In this section I will first review research in robotics, computer vision, and computational linguistics that tries to establish relationships between language and the world. The problem domains addressed span across a wide range including sports, face recognition, navigation, and computer troubleshooting. I will then review work in semantic parsing and language generation that addresses how to transform whole sentences into symbolic, logical representations and vice versa. Our work extends existing techniques in this area to deal with ambiguity in the training data.

2.1 Connecting Language and the World

As Deb Roy discusses in his theoretical framework for grounding language (Roy, 2005), the meaning of language can be divided into two types: referential and functional. Referential meanings talk about objects and events in the world. On the other hand, functional meanings aim to achieve some actions in the world. Most of the existing work focuses on learning referential meanings, or learning how to describe the world with words. However, some more recent work has explored using language as a guide to help accomplish certain tasks. Our existing work on the sportscasting task deals with referential meaning while the proposed work on the navigation task explores functional meaning. In order to learn these meanings however, we need to first align the language with the corresponding perception or knowledge. This is one of the core problems of my research.

2.1.1 Referential Meanings

Referential meanings are useful both for describing the world and for understanding descriptions of the world. The former allows a computer system to automatically generate linguistic reports about its knowledge and perceptions. The latter helps a computer system understand a visual scene from the descriptions.

One of the most ambitious end-to-end visually grounded system scene description system was VITRA (Herzog & Wazinski, 1994) which commented on traffic scenes and soccer matches. The system first transforms raw visual data into geometrical representations. Afterward, a set of cognitive rules extract spatial relations and interesting motion events from those representations. Presumed intentions, plans, and plan interactions between the agents are also extracted based on domain-specific knowledge. However, since their system is hand-coded it cannot be adapted easily to new domains.

Srihari and Burhans used captions accompanying photo to help identify people and objects (Srihari & Burhans, 1994). They introduced the idea of visual semantics, a theory of extracting visual information and constraints from accompanying text. For example, by using caption information, the system can determine the spatial relationship between the entities mentioned, the likely size and shape of the object of interest, and whether the entity is natural or artificial. However, their system also used a lot of hand-coded knowledge.

The earliest work on *learning* word meanings from perception is by Siskind (Siskind, 1996). He described a method of solving referential uncertainty in the language acquisition task. By noticing the same perceptions when the same word is uttered, the meaning of words can eventually be learned. His technique, however, concentrated on word learning and does not address the problem of composing meanings together in a coherent way.

Robotics and computer vision researchers have since then worked on inferring grounded meanings of individual words or short referring expressions from perceptual context, e.g. (Roy, 2002; Bailey et al., 1997; Barnard et al., 2003; Yu & Ballard, 2004). However, the complexity of the natural language used in this existing work is very restrictive, many of the systems use pre-coded knowledge of the language, and almost all use static images to learn language describing objects and their relations, and cannot learn language describing actions. The most sophisticated grammatical formalism used to learn syntax in this work is a finite-state hidden-Markov model. By contrast, our work exploits the latest techniques in statistical context-free grammars and syntax-based statistical machine translation that can handle more of the complexities of real natural language.

More recently, Gold and Scassellati built a system called TWIG that uses existing language knowledge to help it learn the meaning of new words (Gold & Scassellati, 2007). The robot uses partial parses to focus its attention on possible meanings of new words. By playing a game of catch, the robot was able to learn the meaning of "you" and "me" as well as "am" and "are" as identity relations.

Deb Roy has also done a lot of work on integrating computer vision and computational linguistics. He built several robotic systems that performs language grounding in the perceptual world (Roy & Pentland, 2002; Roy, 2001). However, since these systems deal with raw sensory data, they are limited to basic visual features such as color and shape.

There has also been a number of work on learning from captions that accompany pictures or videos (Satoh, Nakamura, & Kanade, 1997; Berg, Berg, Edwards, & Forsyth, 2004). This area is of particular interest because there exists a large amount of data in this form on the web and from television. Satoh et al. built a system to detect faces in newscast. However, their language model uses simple manual rules to determine the most likely entity in the picture (Satoh et al., 1997). Berg et al. used a more elaborate language model to help cluster faces with names. Instead of manual rules, they estimate the likelihood of an entity appearing in a picture by examining its context. The probability of the entity appearing given its context is estimated from the data. However, these systems only aim to extract names from the captions rather than understanding the semantics.

Some recent work on video retrieval has focused on learning to recognize events in sports videos and connecting them to English words appearing in accompanying closed captions text (Fleischman & Roy, 2007; Gupta & Mooney, 2009). However, this work only learns the connection between individual words and video events and does not learn to describe events using full grammatical sentences.

2.1.2 Functional Meanings

Instead of just describing the world, functional meanings are used for instructing actions in the world. Recently, there has been some work that uses language to aid computer agents in completing tasks. For example, Kruijff et al. designed a robot that uses human linguistic inputs to help it build a topological map (Kruijff, Zender, Jensfelt, & Christensen, 2007). The robot also asks the human to clarify uncertainties such as whether there is a door in the nearby vicinity. The system uses hand-built combinatory categorial grammar to parse the language. This is feasible since the robot's dialogues are confined to navigation tasks.

In addition to navigation tasks, there has also been work on learning to perform more complex tasks such as puzzle solving in computer video games (Gorniak & Roy, 2005). In this work, players cooperate and communicate with each in order to accomplish a certain task. The players not only have to navigate to certain locations, they also have to perform actions such as pushing levers and lighting torches in the correct sequence in order to complete the task. However, this work uses an existing statistical parser, and does not learn the syntax and semantics of the language from the perceptual environment alone.

There has also been some interest in learning how to interpret English instructions describing how to use a particular website or perform other computer tasks (Branavan, Chen, Zettlemoyer, & Barzilay, 2009; Lau, Drews, & Nichols, 2009). These systems learn to predict the correct computer action (pressing a button, choosing a menu item, typing into a text field, etc.) corresponding to each step in the instructions. However, these system exploit language-specific knowledge including direct matches between words in the natural language instructions and words that appear in the computer environment in order to establish a connection. Our approach assumes no prior information establishing connections between the language input and the symbolic representation describing the environment and is therefore language independent.

2.1.3 Alignment

One of the core problems our work addresses is matching sentences to facts in the world to which they refer. Some recent projects aimed to align text from English summaries of American football games with database records that contain statistics and events about the game (Snyder & Barzilay, 2007; Liang, Jordan, & Klein, 2009). However, Snyder and Barzilay use a supervised approach that requires annotating the correct correspondences between the text and the semantic representations. On the other hand, Liang *et al.* have developed an unsupervised approach using a generative model to solve the alignment problem. They also demonstrated improved results on matching sentences and events for our RoboCup sportscasting data. However, their work does not address semantic parsing or language generation.

Unlike our project, none of the work referenced above learns to map structures in language to structures in perception (real or simulated) without prior knowledge about the language. Most of the work treats language as individual tokens or features that can be used to classify objects or decide actions. The few systems that utilized statistical parsing techniques did not learn parsers from the data but rather used existing parsers. In contrast, our goal is to map parts of a sentence into logical representations that can then be composed together to form a complete representation of the entire sentence. We can then use this mapping to verbalize knowledge of the computer system into natural language or interpret natural language into a logical language that the computer can perform inference on.

2.2 Semantic Parsing and Language Generation

The goal of our project can be formally described as learning how to build semantic parsers and language generators from perceptual data. Semantic parsers map natural language (NL) sentences to *meaning representations* (MRs) in some formal logical language. On the other hand, language generator performs the reverse mapping (MRs to NL). Existing work has focused on learning from supervised corpora in which each sentence is manually annotated with its correct MR (Mooney, 2007; Zettlemoyer & Collins, 2007; Lu et al., 2008). Such human annotated corpora are expensive and difficult to produce, limiting the utility of this approach. Kate and Mooney (2007) introduced an extension to one such such system, KRISP (Kate & Mooney, 2006), so that it can learn from ambiguous training data that requires little or no human annotation effort. However, their system was unable to *generate* language which is required for our tasks. Thus, we enhanced another system called WASP (Wong & Mooney, 2006) that is capable of language generation as well as semantic parsing in a similar manner to allow it to learn from ambiguous supervision. We briefly describe these systems below. All of them assume they have access to a formal deterministic *context-free grammar* (CFG) that defines the formal *meaning representation language* (MRL). Since MRLs are formal computer-interpretable languages, such grammars are usually readily available.

2.2.1 KRISP and KRISPER

KRISP (Kernel-based Robust Interpretation for Semantic Parsing) (Kate & Mooney, 2006) uses *support vector machines* (SVMs) with string kernels to build semantic parsers. SVMs are state-of-the-art machine learning methods that learn maximum-margin separators to prevent over-fitting in very high-dimensional data such as natural language text (Joachims, 1998). They can be extended to non-linear separators and non-vector data by exploiting *kernels* that implicitly create an even higher dimensional space in which complex data is (nearly) linearly separable (Shawe-Taylor & Cristianini, 2004). Recently, kernels over strings and trees have been effectively applied to a variety of problems in text learning and NLP (Lodhi, Saunders, Shawe-Taylor, Cristianini, & Watkins, 2002; Zelenko, Aone, & Richardella, 2003; Collins, 2002; Bunescu & Mooney, 2005). In particular, KRISP uses the string kernel introduced by Lodhi *et al.* (2002) to classify substrings in a NL sentence.

First, KRISP learns classifiers that recognize when a word or phrase in an NL sentence indicates that a particular concept in the MRL should be introduced into its MR. It uses production rules in the MRL grammar to represent semantic concepts, and it learns classifiers for each production that classify NL substrings as indicative of that production or not. When semantically parsing a sentence, each classifier estimates the probability of each production covering different substrings of the sentence. This information is then used to compositionally build a complete MR for the sentence. Given the partial matching provided by string kernels and the over-fitting prevention provided by SVMs, KRISP has been experimentally shown to be particularly robust to noisy training data (Kate & Mooney, 2006).

KRISPER (Kate & Mooney, 2007) is an extension to KRISP that handles ambiguous training data in which each sentence is annotated with a *set* of potential MRs, only one of which is correct. Psuedocode for KRISPER is shown in Algorithm 1. It employs an iterative approach analogous to *expectation maximization* (EM) (Dempster, Laird, & Rubin, 1977) that improves upon the selection of the correct NL–MR pairs in each iteration. In the first iteration (lines 3-9), it assumes that all of the MRs paired with a sentence are correct and trains KRISP with the resulting noisy supervision. In subsequent iterations (lines 11-27), KRISPER uses the currently trained parser to score each potential NL–MR pair, selects the most likely MR for each sentence, and retrains the parser on the resulting disambiguated supervised data. In this manner, KRISPER is able to learn from the type of weak supervision expected for a grounded language learner exposed only to sentences in ambiguous contexts. However, the system has previously only been tested on artificially corrupted or generated data.

2.2.2 WASP and WASP^{-1}

WASP (Word-Alignment-based Semantic Parsing) (Wong & Mooney, 2006) uses state-of-the-art *statistical machine translation* (SMT) techniques (Brown, Cocke, Della Pietra, Della Pietra, Jelinek, Lafferty, Mercer, & Roossin, 1990; Yamada & Knight, 2001; Chiang, 2005) to learn semantic parsers. SMT methods learn effective machine translators by training on *parallel corpora* consisting of human translations of documents into one or more alternative natural languages. The resulting translators are typically significantly more effective than manually developed systems and SMT has become the dominant approach to machine translation. WASP adapted such methods to learn to translate from NL to MRL rather than from one NL to another.

First, an SMT *word alignment* system, GIZA++ (Och & Ney, 2003; Brown, Della Pietra, Della Pietra, & Mercer, 1993), is used to acquire a bilingual lexicon consisting of NL substrings coupled with their translations in the target MRL. As formal languages, MRLs frequently contain many purely syntactic tokens such as parentheses or brackets which are difficult to align with words in NL. Consequently, WASP aligns words in the NL with productions of the MRL grammar used in the parse of the corresponding MR. Therefore, GIZA++ is used to produce an N to 1 alignment between the words in the NL sentence and a sequence of MRL productions corresponding to a top-down left-most derivation of the corresponding MR.

Complete MRs are then formed by combining these NL substrings and their translations using a grammatical framework called *synchronous CFG* (SCFG) (Aho & Ullman, 1972), which forms the basis of most

Algorithm 1 KRISPER

```
input sentences $S$ and their associated sets of meaning representations $MR(s)$
output $BestExamplesSet$, a set of NL-MR pairs,
    $SemanticModel$, a KRISP semantic parser

main
  //Initial training loop
  for sentence $s_i \in S$ do
    for meaning representation $m_j \in MR(s_i)$ do
      add $(s_i, m_j)$ to $InitialTrainingSet$
    end for
  end for
  $SemanticModel$ = Train($InitialTrainingSet$)

  //Iterative retraining
  repeat
    for sentence $s_i \in S$ do
      for meaning representation $m_j \in MR(s_i)$ do
        $m_j.score = Evaluate(s_i, m_j, SemanticModel)$
      end for
    end for
    $BestExampleSet \leftarrow$ The set of consistent examples $T = \{(s,m) | s \in S, m \in MR(s)\}$ such that
      $\sum_T m.score$ is maximized
    $SemanticModel = Train(BestExamplesSet)$
  until Convergence or MAX_ITER reached
end main

function Train($TrainingExamples$)
  Train KRISP on the unambiguous $TrainingExamples$
  return The trained KRISP semantic parser
end function

function Evaluate($s$, $m$, $SemanticModel$)
  Use the KRISP semantic parser $SemanticModel$ to find a derivation of meaning representation $m$ from
    sentence $s$
  return The parsing score
end function
```

existing *syntax-based* SMT (Yamada & Knight, 2001; Chiang, 2005). In an SCFG, the right hand side of each production rule contains *two* strings, in this case one in NL and the other in MRL. Derivations of the SCFG simultaneously produce NL sentences and their corresponding MRs. The bilingual lexicon acquired from word alignments over the training data is used to construct a set of SCFG production rules. A probabilistic parser is then produced by training a maximum-entropy model using EM to learn parameters for each of these SCFG productions, similar to the methods used in (Riezler, Prescher, Kuhn, & Johnson, 2000; Zettlemoyer & Collins, 2005). To translate a novel NL sentence into its MR, a probabilistic chart parser (Stolcke, 1995) is used to find the most probable synchronous derivation that generates the given NL, and the corresponding MR generated by this derivation is returned.

Since SCFGs are symmetric, they can be used to *generate* NL from MR as well as parse NL into MR (Wong & Mooney, 2007). This allows the same learned grammar to be used for both parsing and generation, an elegant property that has important advantages (Shieber, 1988). The generation system, WASP^{-1}, uses a *noisy-channel model* (Brown et al., 1990):

$$\arg\max_{\mathbf{e}} \Pr(\mathbf{e}|\mathbf{f}) = \arg\max_{\mathbf{e}} \Pr(\mathbf{e}) \Pr(\mathbf{f}|\mathbf{e}) \quad (1)$$

Where **e** refers to the NL string generated for a given input MR, **f**. $\Pr(\mathbf{e})$ is the *language model*, and $\Pr(\mathbf{f}|\mathbf{e})$ is the *parsing model* provided by WASP's learned SCFG. The generation task is to find a sentence **e** such that (1) **e** is a good sentence a priori, and (2) its meaning is the same as the input MR. For the language model, WASP^{-1} uses a standard *n*-gram model, which is useful in ranking candidate generated sentences (Knight & Hatzivassiloglou, 1995).

Since both WASP and WASP^{-1} learn the same underlying SCFG rules, we will refer to both system jointly as WASP in the rest of this proposal.

2.2.3 Strategic Generation

Prior to transforming a MR into a NL utterance, we must first decide what to say. This is referred to as the task of *strategic generation* or content selection. There has been some recent work on learning strategic language generation using reinforcement learning (Zaragoza & Li, 2005). They use a setting similar to our navigation task in a video game environment where the speaker must aid the listener in reaching the destination while avoiding obstacles. They repeatedly played the game to find an optimal strategy that conveyed the most pertinent information while minimizing the amount of messages. However, they do not consider the case where many different messages can be equally informative but some are preferred more by humans.

In addition, there has also been work on performing strategic generation as a collective task (Barzilay & Lapata, 2005). By considering all the strategic generation decisions jointly, they capture dependencies between each uttered items. This creates more consistent overall output and aligns better with how humans perform this task.

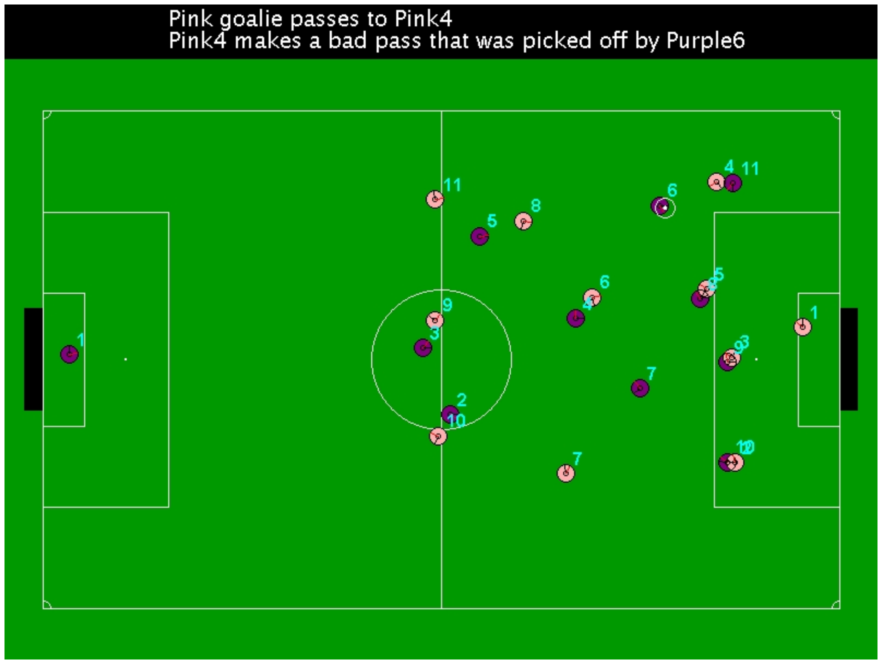

Pink goalie passes to Pink4
Pink4 makes a bad pass that was picked off by Purple6

Figure 1: Screenshot of our commentator system

3 Completed Research

As a step toward solving the problem of language grounding using perceptual context, we present a system that can describe events in a simulated soccer game by learning only from sample language commentaries paired with traces of simulated activity without any language-specific prior knowledge. A screenshot of our system with generated commentary is shown in Figure 1.

Our commentator system learns to semantically interpret and generate language in the RoboCup soccer domain by observing an on-going commentary of the game paired with the evolving simulator state. Specifically, we use the RoboCup simulator (Chen, Foroughi, Heintz, Kapetanakis, Kostiadis, Kummeneje, Noda, Obst, Riley, Steffens, Wang, & Yin, 2003) which provides a fairly detailed physical simulation of robot soccer. While several groups have constructed RoboCup commentator systems (André, Binsted, Tanaka-Ishii, Luke, Herzog, & Rist, 2000) that provide a textual NL transcript of the simulated game, their systems use manually-developed templates and are not based on learning. By exploiting existing techniques for abstracting a symbolic description of the activity on the field from the detailed states of the physical simulator (André et al., 2000), we obtain a pairing of natural language with a symbolic description of the perceptual context in which it was uttered. However, such training data is highly ambiguous because each comment usually co-occurs with several events in the game. We integrate and enhance existing methods for learning semantic parsers and NL generators (Kate & Mooney, 2007; Wong & Mooney, 2007) in order to learn to understand and generate language from such ambiguous training data. We also develop a system that, from the same ambiguous training data, learns which events are worth describing, so that it can also perform *strategic generation*, that is, deciding *what* to say as well as how to say it (*tactical generation*).

Experiments on test data (annotated for evaluation purposes only) demonstrate that the system learns to accurately semantically parse sentences, generate sentences, and decide which events to describe. Finally, subjective human evaluation of commentated game clips demonstrate that the system generates sportscasts that are in some cases similar in quality to those produced by humans. While the results presented in this proposal is on the English data we collected, we have also tested our system on Korean data and obtained similar results.

We describe the sportscasting task and the data we collected to train and test our system in more detail in Section 3.1. We then present our basic system and associated experimental results in Section 3.2 and Section 3.3 respectively. We discuss extensions to the system by initializing it with data disambiguated by a generative model in Section 3.4 and removing "superfluous" sentences that do not refer to any extracted event in Section 3.5. Finally, we show the results of human evaluation of our overall system in Section 3.6.

3.1 Sportscasting Task

The goal of the sportscasting task is to produce a play-by-play call of a RoboCup simulation game. To achieve this, the system must first perform strategic generation and choose the relevant events to describe. Once it has chosen the events, it then has to form natural language descriptions of the events. Since our main focus is on developing the mapping between language and semantics, we did not consider the overall coherence of the sportscast while developing our system. We treated each utterance decision independently and did not try to add any color commentaries (e.g. commentaries used to fill in any time when play is not in progress, examples include expert analysis and statistics of the players and teams) that is usually part of a sportscast.

To train and test our system, we assembled human-commentated soccer games from the RoboCup simulation league (www.robocup.org). Since our focus is language learning and not computer vision, we chose to use simulated games instead of real game videos to simplify the extraction of perceptual information. Symbolic representations of game events were automatically extracted from the simulator traces by a rule-based system. The extracted events mainly involve actions with the ball, such as kicking and passing, but also include other game information such as whether the current playmode is kickoff, offside, or corner kick. The events are represented as atomic formulas in predicate logic with timestamps. These logical facts constitute the requisite MRs, and we manually developed a simple CFG for this formal semantic language.

For the NL portion of the data, we had humans commentate games while watching them on the simulator. The commentators typed their comments into a text box, which were recorded with a timestamp. To construct the final ambiguous training data, we paired each comment with all of the events that occurred five seconds or less before the comment was made. A sample set of the ambiguous training data is shown in Figure 2. The edges connect sentences to events to which they might refer. Note that the use of English words for predicates and constants in the MRs is for human readability only, the system treats these as arbitrary conceptual tokens and must learn their connection to English words.

Two commentators annotated a total of four games, namely, the finals for the RoboCup simulation league for each year from 2001 to 2004. Summary statistics about the data are shown in Table 1. The 2001 final has almost twice the number of events as the other games because it went into double overtime.

For evaluation purposes only, a gold-standard matching was produced by examining each comment manually and selecting the correct MR if it existed. The matching is only approximate because sometimes the comments contain more information than present in the MRs. For example, a comment might describe the location and the length of a pass while the MR captures only the participants of a pass. The bold lines in Figure 2 indicate the annotated correct matches in our sample data. Notice some sentences do not have correct matches (about one fifth of the data). For example, the sentence "Purple team is very sloppy today" in Figure 2 cannot be represented in our MRL and consequently does not have a corresponding correct MR. Also, in the case of the sentence "Pink11 makes a long pass to Pink8" in Figure 2, the correct MR falls outside of the 5-second window. For each game, Table 1 shows the total number of NL sentences, the

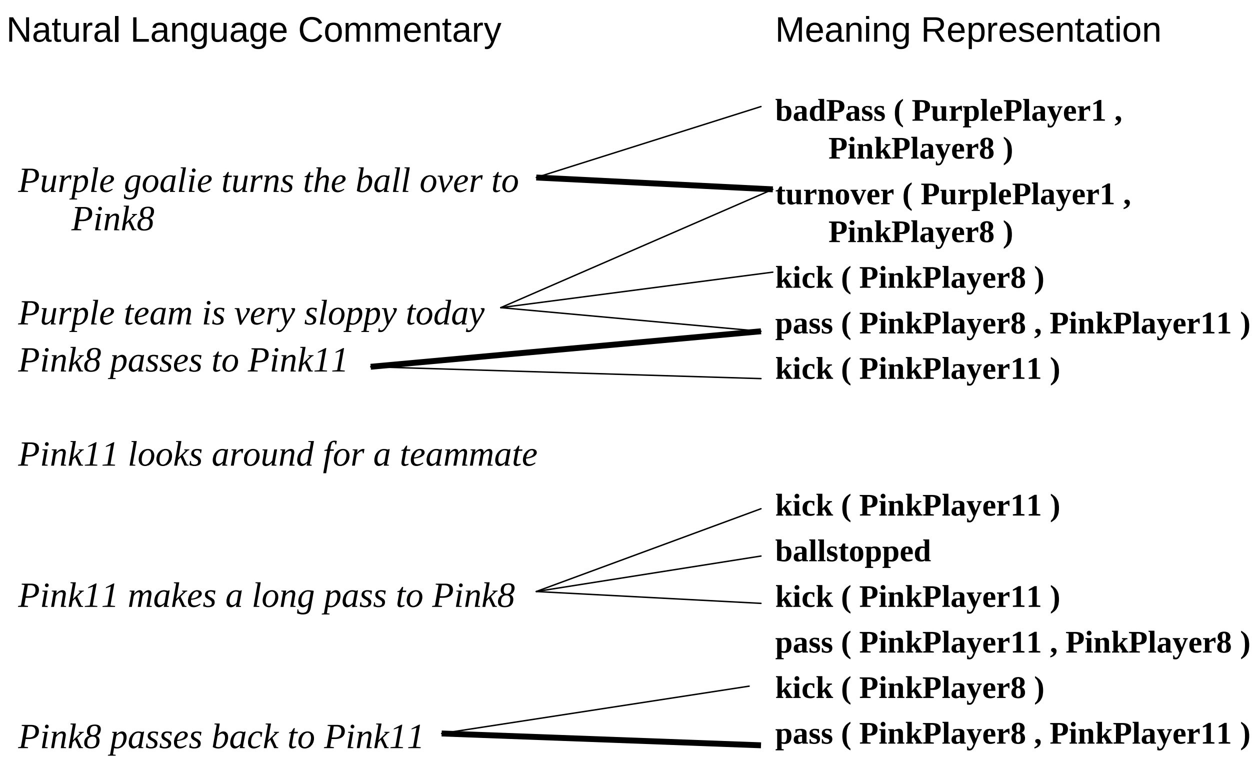

Natural Language Commentary
Meaning Representation
Purple goalie turns the ball over to Pink8
Purple team is very sloppy today
Pink8 passes to Pink11
Pink11 looks around for a teammate
Pink11 makes a long pass to Pink8
Pink8 passes back to Pink11
badPass (PurplePlayer1 , PinkPlayer8)
turnover (PurplePlayer1 , PinkPlayer8)
kick (PinkPlayer8)
pass (PinkPlayer8 , PinkPlayer11)
kick (PinkPlayer11)
kick (PinkPlayer11)
ballstopped
kick (PinkPlayer11)
pass (PinkPlayer11 , PinkPlayer8)
kick (PinkPlayer8)
pass (PinkPlayer8 , PinkPlayer11)

Figure 2: Sample trace of ambiguous training data

		Number of comments			Events per comment		
	Number of events	Total	Have MRs	Have Correct MR	Max	Average	Std. Dev.
2001 final	4003	722	671	520	9	2.235	1.641
2002 final	2223	514	458	376	10	2.403	1.653
2003 final	2113	410	397	320	12	2.849	2.051
2004 final	2318	390	342	323	9	2.729	1.697

Table 1: Statistics about the dataset

number of these that have at least one recent extracted event to which it *could* refer, and the number of these that actually *do* refer to one of these recent extracted events. The maximum, average, and standard deviation for the number of recent events paired with each comment are also given.

3.2 Learning Language from Ambiguous Supervision

While existing systems are capable of solving parts of the sportscasting problem, none of them are able to perform the whole task. We need a system that can learn how to do both tactical and strategic generation from ambiguous training data. We first present three systems that combine the ideas from KRISPER and WASP to learn tactical generators from ambiguous supervision. We then present a novel algorithm for learning strategic generation by exploiting occurrences of events without accompanying comments.

All three tactical generation systems introduced here are based on extensions to WASP, our underlying language learner. The main problem we need to solve is to disambiguate the training data so that we can train WASP as before to create a language generator.

3.2.1 WASPER

The first system is an extension of WASP in a manner similar to how KRISP was extended to create KRISPER. It uses an EM-like retraining to handle ambiguously annotated data, resulting in a system we call WASPER. In general, any system that learns semantic parsers can be extended to handle ambiguous data as long as it can produce confidence levels for given NL–MR pairs. The pseudocode for WASPER is shown in Algorithm 2. The only difference compared to the KRISPER pseudocode is that we now use a WASP semantic parser instead of a KRISP parser. Also, we produce a WASP language generator as well which is the desired final output for our task.

Algorithm 2 WASPER

```
input  sentences $S$ and their associated sets of meaning representations $MR(s)$
output  BestExamplesSet, a set NL-MR pairs,
    SemanticModel, a WASP semantic parser/language generator
 1: main
 2:     same as Algorithm 1
 3: end main
 4:
 5: function Train(TrainingExamples)
 6:     Train WASP on the unambiguous TrainingExamples
 7:     return The trained WASP semantic parser/language generator
 8: end function
 9:
10: function Evaluate($s$, $m$, SemanticModel)
11:     Use the WASP semantic parser in SemanticModel to find a derivation of meaning representation $m$
        from sentence $s$
12:     return The parsing score
13: end function
```

3.2.2 KRISPER-WASP

KRISP has been shown to be quite robust at handling noisy training data (Kate & Mooney, 2006). This is important when training on the very noisy training data used to initialize the parser in KRISPER's first iteration. However, KRISPER cannot learn a language generator, which is necessary for our sportscasting task. As a result, we create a new system called KRISPER-WASP that is theoretically both good at disambiguating the training data and capable of generation. We first use KRISPER to train on the ambiguous data and produce a disambiguated training set by using its prediction for the most likely MR for each sentence. This unambiguous training set is then used to train WASP to produce both a parser and a generator.

3.2.3 WASPER-GEN

In both KRISPER and WASPER, the criterion for selecting the best NL–MR pairs during retraining is based on maximizing the probability of parsing a sentence into a particular MR. However, since WASPER is capable of both parsing and generation, we could alternatively select the best NL–MR pairs by evaluating how likely it is to *generate* the sentence from a particular MR. Thus, we built another version of WASPER called WASPER-GEN that disambiguates the training data in order to maximize the performance of *generation* rather than parsing. The pseudocode is shown in Algorithm 3. The algorithm is the same as WASPER except for the evaluation function. It uses a generation-based score rather than a parsing-based score to select the best NL–MR pairs.

Specifically, an NL–MR pair (s, m) is scored by using the current trained generator to generate an NL sentence for m and then comparing the generated sentence to s by computing the NIST score (lines 9-12). NIST score is a machine translation (MT) evaluation metric that measures the precision of a translation in terms of the proportion of n-grams it shares with a human translation (Doddington, 2002). It has also been used to evaluate NL generation. Another popular MT metric is BLEU score (Papineni, Roukos, Ward, & Zhu, 2002), but it is inadequate for our purpose since we are comparing one short sentence to another instead of comparing whole documents. BLEU score computes the geometric mean of the n-gram precision for each value of n, which means the score is 0 if a matching n-gram is not found for *every* value of n. In the common setting in which the maximum n is 4, any two sentences that do not have a matching 4-gram would receive a BLEU score of 0. Consequently, BLEU score is unable to distinguish the quality of most of our generated sentences since they are fairly short. In contrast, NIST uses an additive score and avoids this problem.

3.2.4 Strategic Generation

A language generator alone is not enough to produce a sportscast. A sportscasting system must also choose which events to describe. Thus, we developed a simple method for learning strategic generation. For each event type (i.e. for each predicate like `pass`, or `goal`), the system uses the training data to estimate the probability that it is mentioned by the sportscaster. Given the gold-standard NL–MR matches, this probability is easy to estimate; however, the learner does not know the correct matching. Instead, the system must estimate the probabilities from the ambiguous training data. We compare two basic methods for estimating these probabilities.

The first method uses the *inferred* NL–MR matching produced by the language-learning system. The probability of commenting on each event type, e_i, is estimated as the percentage of events of type e_i that have been matched to *some* NL sentence.

The second method, which we call Iterative Generation Strategy Learning (IGSL), uses a variant of EM,

Algorithm 3 WASPER-GEN

```
input  sentences $S$ and their associated sets of meaning representations $MR(s)$
output  $BestExamplesSet$, a set of NL-MR pairs,
    $SemanticModel$, a WASP semantic parser/language generator
1: main
2:     same as Algorithm 1
3: end main
4:
5: function Train($TrainingExamples$)
6:     same as Algorithm 2
7: end function
8:
9: function Evaluate($s$, $m$, $SemanticModel$)
10:     $GeneratedSentence \leftarrow$ Use the WASP language generator in $SemanticModel$ to produce a sentence
        from the meaning representation $m$
11:     return The NIST score between $GeneratedSentence$ and $s$
12: end function
```

treating the matching assignments as hidden variables, initializing each match with a prior probability, and iterating to improve the probability estimates of commenting on each event type. Unlike the first method, IGSL uses information about MRs not explicitly associated with any sentence in training. Algorithm 4 shows the pseudocode. The main loop alternates between estimating the probability of each NL–MR matching (line 6) and the prior probability that an event type is mentioned by a human commentator (line 9). In the first iteration, each NL–MR match is assigned a probability inversely proportional to the amount of ambiguity associated with the sentence ($\sum_{e \in Event(s)} Pr(e) = |Event(s)|$). For example, a sentence associated with five possible MRs will assign each match a probability of $\frac{1}{5}$. The prior probability of mentioning an event type is then estimated as the average probability assigned to instances of this event type. Notice this process does not always guarantee a proper probability since a MR can be associated with multiple sentences. Thus, we limit the probability to be at most one. In the subsequent iterations, the probabilities of the NL–MR matchings are updated according to these new priors. We assign each match the prior probability of its event type normalized across all the associated MRs of the NL sentence. We then update the priors for each event type as before using the new estimated probabilities for the matchings. This process is repeated until the probabilities converge or a pre-specified number of iterations has occurred.

To generate a sportscast, we first use the learned probabilities to determine which events to describe. For each time step, we only consider commenting on the event with the highest probability. The system then stochastically decides whether to generate a comment for this event based on the estimated probability for its event type.

3.3 Experimental Evaluation

This section presents experimental results on the RoboCup data for our system. We first present evaluation of the tactical generation systems and then the strategic generation component.

For the tactical generation evaluation, we compare the performances of four systems: KRISPER, WASPER, KRISPER-WASP, and WASPER-GEN. To better gauge the effect of accurate ambiguity resolution, we include results of unmodified WASP. Since WASP requires unambiguous training data, we randomly

Algorithm 4 Iterative Generation Strategy Learning

```
input event types $E = \{e_1, ..., e_n\}$, the number of occurrences of each event type $TotalCount(e_i)$ in the
    entire game trace, sentences $S$ and the event types of their associated meaning representations $Event(s)$
output probabilities of commenting on each event type $Pr(e_i)$
1: Initialize all $Pr(e_i) = 1$
2: repeat
3:   for event type $e_i \in E$ do
4:     $MatchCount = 0$
5:     for sentence $s \in S$ do
6:       $ProbOfMatch = \frac{\sum_{e \in Event(s) \wedge e = e_i} Pr(e)}{\sum_{e \in Event(s)} Pr(e)}$
7:       $MatchCount = MatchCount + ProbOfMatch$
8:     end for
9:     $Pr(e_i) = min(\frac{MatchCount}{TotalCount(e_i)}, 1)$ {Ensure proper probabilities}
10:  end for
11: until Convergence or MAX_ITER reached
```

picked a meaning for each sentence from its set of potential MRs. This forms our lower baseline. Finally, we also include the result of WASP trained using the *gold matching* which consists of the correct NL–MR pairs annotated by a human. This represents an upper-bound on what our systems could achieve if they disambiguated the training data perfectly.

We evaluate each system on three tasks: matching, semantic parsing, and tactical generation. The matching task measures how well the systems can disambiguate the training data. The parsing and generation tasks measure how well the systems can translate from NL to MR, and from MR to NL, respectively.

Since there are four games in total, we trained using all possible combinations of one to three games. For matching, we measured the performance on the training data since our goal is to disambiguate this data. For semantic parsing and tactical generation, we tested on the games not used for training. Results were averaged over all train/test combinations. We evaluated matching and parsing using F-measure, the harmonic mean of recall and precision. Precision is the fraction of the system's annotations that are correct. Recall is the fraction of the annotations from the gold-standard that the system correctly produces. Generation is evaluated using BLEU scores which roughly estimates how well the produced sentences match with the target sentences. We treat each game as a whole document to avoid the problem of using BLEU score for sentence-level comparisons mentioned earlier. Also, we increase the number of reference sentences for each MR by using all of the sentences in the test data corresponding to equivalent MRs, e.g. if `pass(PinkPlayer7, PinkPlayer8)` occurs multiple times in the test data, all of the sentences matched to this MR in the gold matching are used as reference sentences for this MR.

For the strategic generation evaluation, we compare the performances of using the inferred matchings and IGSL. We use a similar train/test split as for the semantic parsing and tactical generation tasks. Performance is measured by F-measure of how often the algorithms select the same events to commentate as the humans.

3.3.1 Matching NL and MR

Since handling ambiguous training data is an important aspect of grounded language learning, we first evaluate how well the various systems pick the correct NL–MR pairs. Figure 3 shows the F-measure for identifying the correct set of pairs for the various systems. All of the learning systems perform significantly

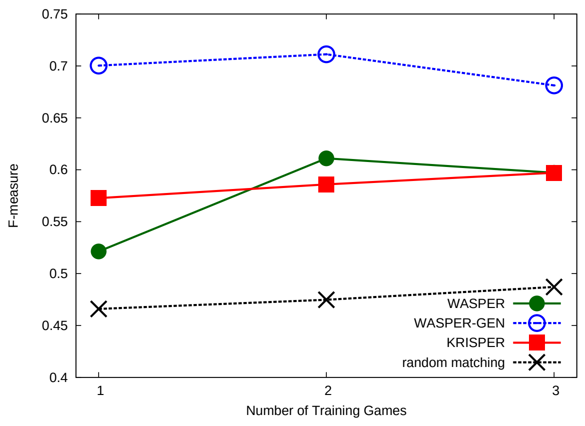


Figure 3: Matching results

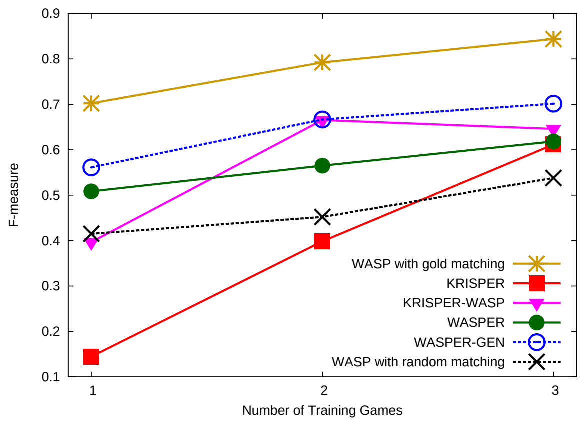


Figure 4: Semantic parsing results

better than random which has a F-measure below 0.5. Across the learning curve, WASPER-GEN is the best system. WASPER also equals or outperforms the previous system KRISPER most of the time.

3.3.2 Semantic Parsing

Next, we present results on the accuracy of the learned semantic parsers. Each trained system is used to parse and produce an MR for each sentence in the test set that has a correct MR in the gold-standard matching. A parse is considered correct if and only if it matches the gold standard exactly. Parsing is a fairly difficult task because there is usually more than one way to describe the same event. For example, "Player1 passes to player2" can refer to the same event as "Player1 kicks the ball to player2." Thus, accurate parsing requires learning all the different ways people describe an event. Synonymy is not limited to verbs. In our data, "Pink1", "PinkG" and "pink goalie" all refer to player1 on the pink team. Since we are not providing the systems with any prior knowledge, they have to learn all these different ways of referring to the same entity.

The parsing results shown in Figure 4 correlate well with the matching results. Systems that did better at disambiguating the training data also did better on parsing because their supervised training data is less noisy. WASPER-GEN again does the best overall. It is interesting to note that systems that use KRISP have

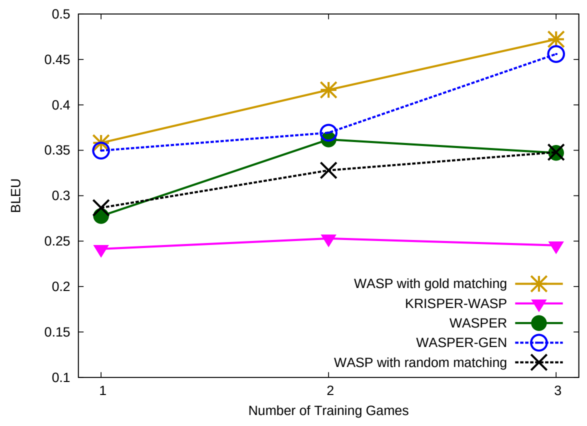


Figure 5: Tactical generation results

distinctive curves. A closer inspection of the results indicates that KRISP is worse at generalizing over different styles of commentating. As mentioned in Section 3.1 the data was commentated by two different people so any split of data in which the same commentator was not included in both the training and testing data resulted in poor performance for KRISP.

3.3.3 Tactical Generation

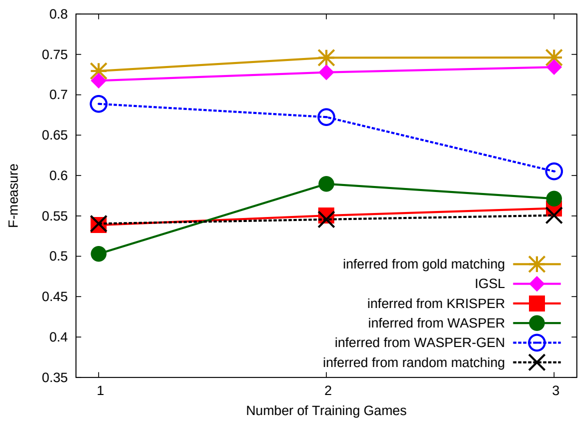
The third evaluation task is tactical generation. All of the WASP-based systems are given each MR in the test set that has a gold-standard matching NL sentence and asked to generate an NL description. The quality of the generated sentence is measured by comparing it to the gold-standard using BLEU scoring.

This task is easier than parsing because the system only needs to learn *one way* to accurately describe an event. This property is reflected in the results, shown in Figure 5, where even the baseline system, WASP with random matching, does fairly well, outperforming KRISPER-WASP. As the number of event types is fairly small, only a relatively small number of correct matchings is required to perform this task well as long as each event type is associated with some correct sentence pattern more often than any other sentence pattern. Thus, the results do not correlate as much to the matching results as semantic parsing does. Instead, it is far more costly to make systematic errors as is the case for WASPER and KRISPER-WASP.

Even though systems such as WASPER and KRISPER-WASP do fairly well at disambiguating the training data, the mistakes they make in selecting the NL–MR pairs often repeat the same basic error. For example, a bad pass event is always followed by a turnover event.[1] If initially the system incorrectly determines that the comment "Player1 turns the ball over to the other team" refers to a bad pass, it will parse the sentence "Player2 turns the ball over to the other team" as a bad pass as well since it just reinforced that connection. Even if the system trains on a correct example where a bad pass is paired with the linguistic input "Player1 made a bad pass", it does not affect the parsing of the first two sentences and does not correct the mistakes. As a result, a bad pass becomes incorrectly associated with the sentence pattern "Someone turns the ball over to the other team."

On the other hand, WASPER-GEN does the best due to the imbalance between the variability of natural language comments and the MRs. While the same MR will typically occur many times in a game, NL

[1]The distinction between bad pass and turnover is that a turnover can occur without any passing, e.g. a player loses control of the ball while dribbling.



(a) English

Figure 6: Strategic generation results

comments are rarely repeated exactly. This leads to the following two performance advantages for WASPER-GEN.

First, WASPER-GEN avoids making the same kind of systematic mistakes as WASPER and KRISPER-WASP. Continuing the previous example, when WASPER-GEN encounters the correct matching for bad pass, it learns to associate bad passes with the correct sentence pattern. When it goes back to those first two incorrect pairings, it will likely correct its mistakes. This is because the same MR bad pass is present in all three examples. Thus, it will slowly move away from the incorrect connections. Of course, parsing and generation are symmetrical processes, so using generation to disambiguate data has its own problems. Namely, it is possible to converge to a point where many events generate the same natural language description. However, since there is much more variability in natural language, it is very unlikely that the same sentence pattern will occur repeatedly, each time associated with different events.

Another performance advantage of WASPER-GEN stems from its evaluation function. Systems like WASPER and KRISPER-WASP that use parsing scores attempt to learn a good translation model for each sentence pattern. On the other hand, WASPER-GEN only tries to learn a good translation model for each MR pattern. Thus, WASPER-GEN is more likely to converge on a good model as there are fewer MR patterns than sentence patterns. However, it can be argued that learning good translation models for each sentence pattern will help in producing more varied commentaries, a quality that is not captured by the BLEU score.

3.3.4 Strategic Generation

The different methods for learning strategic generation are evaluated based on how often the events they describe in the test data coincide with those the human decided to describe. Given all the events that occurred in a game, each algorithms selects a subset to be commentated on. This subset is then compared to the subset of events the human commented on. For the first method, results using the inferred matchings produced by KRISPER, WASPER, and WASPER-GEN as well as the gold and random matching for establishing baselines are all presented in Figure 6. From the graph, it is clear that IGSL outperforms learning from the inferred matchings and actually performs at a level close to using the gold matching. However, it is important to note that we are limiting the potential of learning from the gold matching by using only the predicates to decide whether to talk about an event.

To get a better idea of what the methods are learning, probabilities learned by IGSL and by inferred

event	# occurrences	% commented	IGSL	inferred from WASPER-GEN
ballstopped	5817	1.72×10^{-4}	1.09×10^{-5}	0.016
kick	2122	0.0 33	0.018	0.117
pass	1069	0.999	0.983	0.800
turnover	566	0.214	0.909	0.353
badPass	371	0.429	0.970	0.493

Table 2: Top 5 most frequent events, the % of times they were commented on, and the probabilities learned by the top algorithms for the English data

matchings from WASPER-GEN for the five most frequently occurring events are shown in Table 2. While WASPER-GEN learns fairly good probabilities in general, it does not do as well as IGSL for the most frequent events. This is because IGSL uses occurrences of events that are not associated with any possible comments in its training iterations. Rarely commented events such as *ballstopped* and *kick* often occur without any comments being uttered. Consequently, IGSL assigns low prior probabilities to them which lowers their chances of being matched to any sentences. On the other hand, WASPER-GEN does not use these priors and sometimes incorrectly matches comments to them. Thus, using the inferred matches from WASPER-GEN results in learning higher probabilities of commenting on these rarely commented events.

While all our methods only use the predicates of the MRs to decide whether to comment or not, they do fairly well on the data we collected. In particular, IGSL performs the best, so we use it for strategic generation in the rest of the paper.

3.4 Using a Generative Alignment Model

Recently, Liang *et al.* (2009) developed a generative model that can be used to match natural-language sentences to facts in a corresponding database to which they may refer. As one of their evaluation domains, they used our RoboCup sportscasting data. Their method solves the matching (alignment) problem for our data, but does not address the tasks of semantic parsing or language generation. However, their generative model elegantly integrates simple strategic and tactical language generation models in order to find the overall most probable alignment of sentences and events. They demonstrated improved matching performance on our data, generating more accurate NL–MR pairs than our best system. Thus, we were curious if their results could be used to improve our own systems, which also perform semantic parsing and generation.

The simplest way of utilizing their results is to use the NL–MR pairs produced by their method as supervised data for WASP. We only present results training on three or all four games since they did not run their systems training on lower number of games. As expected, the improved NL–MR pairs from their system did result in improved semantic parsers as can be seen in the results in Table 4. We have included the results from our previously best system (WASPER-GEN) on the last row for comparison. However, language-generation performance was not significantly improved over our previously best results (WASPER-GEN) as shown by the results in Table 5. Once again, a more accurate matching improves parsing but not generation.

In addition to training WASP with their alignment, we can also utilize their output as a better *starting point* for our own systems. Instead of initializing our iterative alignment methods with a model trained on *all* of the ambiguous NL–MR pairs, they can be initialized with the disambiguated NL–MR pairs produced by Liang *et al.*'s system.

Initializing WASPER and WASPER-GEN in this manner improved the matching result compared to our previously best system, WASPER-GEN as shown in Table 3. However, only WASPER was able to improve on the initial matching while WASPER-GEN actually made the matching worse when trained on all data. This is

Method	4-fold cross-validation	all data
Liang et al. (2009)	75.7	80.5
WASPER initialized with Liang et al. (2009)	79.3	80.7
WASPER-GEN initialized with Liang et al. (2009)	75.8	77.5
WASPER-GEN	68.1	n/a

Table 3: English matching results (F1 scores)

Method	F1
WASP with gold matching	84.4
WASP with Liang et al. (2009)	80.3
WASPER initialized with Liang et al. (2009)	79.3
WASPER-GEN initialized with Liang et al. (2009)	77.6
WASPER-GEN	70.2

Table 4: English 4-fold cross-validated parsing results

Method	BLEU
WASP with gold matching	0.472
WASP with Liang et al. (2009)	0.458
WASPER initialized with Liang et al. (2009)	0.460
WASPER-GEN initialized with Liang et al. (2009)	0.441
WASPER-GEN	0.456

Table 5: English 4-fold cross-validated generation results

a reversal of the results from previous sections in which WASPER-GEN consistently outperformed WASPER. The reason for this can be attributed to the nature of using generation scores. Recall that generation is an easier task and that the language generator can perform fairly well even when the data is really noisy. However, its effectiveness also reaches a ceiling sooner as can be observed in Figure 5 where WASPER-GEN already approaches the performance of the upper baseline despite training on data that is more than one quarter noise. Thus, initializing the system with better NL–MR pairs has a more limited effect on improving generation performance. Moreover, the generation evaluation metric is inherently more fickle than that for semantic parsing. This is especially true for our usage where one generated sentence is compared to only one reference sentence and both sentences are usually very short. Compared to the target sentence "Pink goalie kicks the ball to pink4," a perfectly good generated sentence such as "Pink1 passes to pink4" will receive a low generation score and lose against the sentence "Pink goalie blocks the ball." Thus, while WASPER-GEN obtains relatively good performance when the data is very ambiguous, the semantic parser has the potential to help achieve a higher accuracy on the matching task.

Semantic parsing results with initialization are shown in Table 4. Again, both WASPER and WASPER-GEN with initialization beat our previously best system, WASPER-GEN. However, neither of them outperforms simply training WASP with Liang *et al.*'s alignment. This is an exception to our general observation that parsing results are usually consistent with matching results. However, by looking at the upper baseline (training WASP with the gold-standard matching), it is evident that the best results are starting to approach optimal performance. Thus, further improvement in the matching results has less potential impact on improving the parsing results.

Finally, the generation results with initialization are shown in Table 5. Here, the ceiling effect is even more apparent than for semantic parsing. Initializing WASPER-GEN with Liang *et al.*'s alignment actually hurts its performance. On the other hand, WASPER with initialization slightly outperforms our previously best system for this task, WASPER-GEN, as well as slightly improves the results of simply training WASP on the alignment.

Overall, initializing our systems with the alignment output of Liang *et al.*'s generative model improved performance as expected. Starting with a cleaner set of data led to better initial semantic parsers and language generators which led to better end results. Furthermore, by using a strong full semantic parser, WASPER is able to improve on the initial alignment and achieve slightly better generation results. On the other hand, WASPER-GEN seems limited in its ability to identify correct NL–MR pairs and reaches a ceiling sooner.

3.5 Removing Superfluous Comments

So far, we have only discussed how to handle ambiguity in which there are multiple possible MRs for each NL sentence. During training, the previous methods all assume that each NL sentence matches exactly one of the potential MRs. However, some comments are *superfluous*, in the sense that they do not refer to *any* currently extracted event represented in the set of potential MRs. As previously shown in Tables 1, about one fifth of the data are superfluous in this sense.

There are many reasons for superfluous sentences. They occur naturally in language because people do not always talk about the current environment. In our domain, sportscasters often mention past events or more general information about particular teams or players. Moreover, depending on the application, the chosen MRL may not represent all of the things people talk about. For example, our RoboCup MRL cannot represent information about players who are not actively engaged with the ball. Finally, even if a sentence can be represented in the chosen MRL, errors in the perceptual system or an incorrect estimation of when an event occurred can also lead to superfluous sentences. Such perceptual errors can be alleviated to some degree by increasing the size of the window used to capture potential MRs (the previous 5 seconds in our experiments). However, this comes at the cost of increased ambiguity because it associates more MRs with each sentence.

To deal with the problem of superfluous sentences, we can eliminate the lowest-scoring NL–MR pairs (e.g. lowest parsing scores for WASPER or lowest NIST scores for WASPER-GEN). However, in order to set the pruning threshold, we need to automatically estimate the *amount* of superfluous commentary in the absence of supervised data. Notice that while this problem looks similar to the strategic generation problem (estimating how likely an MR participates in a correct matching as opposed to how likely an NL sentence participates in a correct matching), the approaches used there cannot be applied. First, we cannot use the matches inferred by the existing systems to estimate the fraction of superfluous comments since the current systems match every sentence to some MR. It is also difficult to develop an algorithm similar to IGSL due to the imbalance between NL sentences and MRs. Since there are many more MRs, there are more examples of events occurring without commentaries than vice versa.

We propose using a form of internal (i.e. within the training set) cross validation to estimate the rate of superfluous comments. While this algorithm can be used in conjunction with any of our systems, we chose to implement it for KRISPER which trains much faster than our other systems. This makes it more tractable to train many different semantic parsers and choose the best one. The basic idea is to use part of the ambiguous training data to estimate the accuracy of a semantic parser even though we do not know the correct matchings. Assuming a reasonable superfluous sentence rate, we know that most of the time the correct MR is contained in the set of MRs associated with an NL sentence. Thus, we assume that a semantic

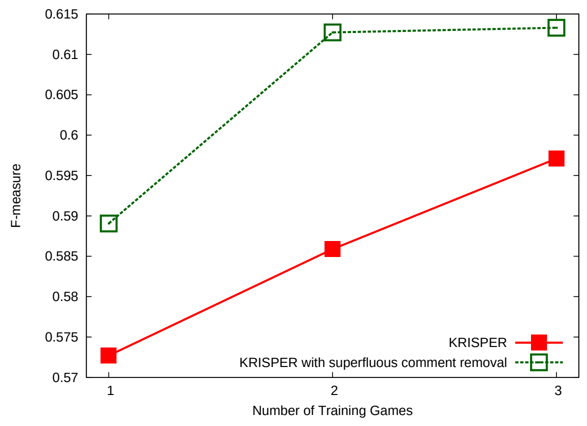


Figure 7: Matching results after removing superfluous sentences

parser that parses an NL sentence into one of the MRs associated with it is better than one that parses it into an MR not in the set. With this approach to estimating accuracy, we can evaluate semantic parsers learned using various thresholds and pick the best one. The algorithm is briefly summarized in the following steps:

1. Split the training set into an *internal training set* and an *internal validation set*.
2. Train KRISPER *N* times on the *internal training set* using *N* different threshold values (eliminating the lowest scoring NL–MR pairs below the threshold in each retraining iteration in Algorithm 1).
3. Test the *N* semantic parsers on the *internal validation set* and determine which parser is able to parse the largest number of sentences into one of their potential MRs.
4. Use the threshold value that produced the best parser in the previous step to train a final parser on the complete original training set.

We evaluated the effect of removing superfluous sentences on these tasks: matching, semantic parsing, and tactical generation. We present results for both KRISPER and KRISPER-WASP. For matching, we only show results for KRISPER because it is responsible for disambiguating the training data for both systems (so KRISPER-WASP's results are the same). For generation, we only show results for KRISPER-WASP, since KRISPER cannot perform generation.

The matching results shown in Figure 7 demonstrate that removing superfluous sentences does improve the performance, although the difference is small in absolute terms.

The parsing results shown in Figure 8 indicate that removing superfluous sentences usually improves the accuracy of both KRISPER and KRISPER-WASP a bit. As we have generally observed, the parsing results are consistent with the matching results.

Finally, the tactical generation results shown in Figure 9 suggest that removing superfluous comments actually decreases performance somewhat. Once again, a potential explanation is that generation is less sensitive to noisy training data. While removing superfluous comments improves the purity of the training data, it also removes potentially useful examples. Consequently, the system does not learn how to generate sentences that were removed from the data. Overall, for generation, the advantage of having cleaner disambiguated training data is apparently outweighed by the loss of data.

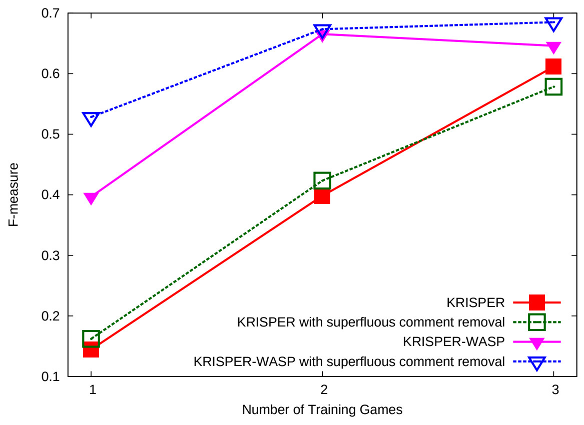


Figure 8: Semantic parsing results after removing superfluous sentences

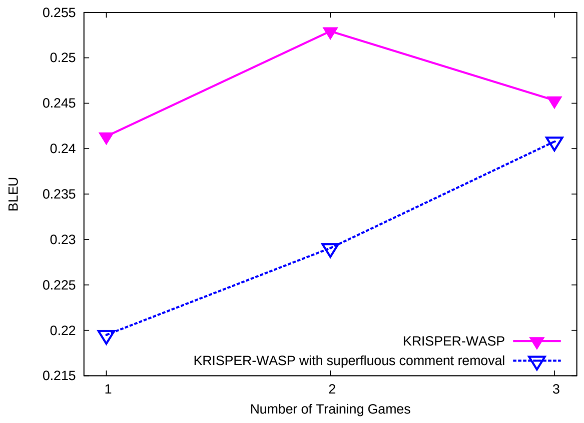


Figure 9: Tactical generation results after removing superfluous sentences

3.6 Human Subjective Evaluation

At best, automatic evaluation of generation is an imperfect approximation of human assessment. Moreover, automatically evaluating the quality of an entire generated sportscast is even more difficult. Consequently, we used Amazon's Mechanical Turk to collect human judgements of the produced sportscasts. Each human judge was shown three clips of simulated game video in one sitting. There were 8 video clips total. The 8 clips use 4 game segments of 4 minutes each, one from each of the four games (2001-2004 RoboCup finals). Each of the 4 game segments is commentated once by a human and once by our system. We use IGSL to determine the events to comment on and WASPER-GEN (our best performing system for tactical generation) to produce the commentaries. To make the commentaries more varied, we took the top 5 outputs from WASPER-GEN and chose one stochastically weighted by their scores. The system was always trained on three games, leaving out the game from which the test segment was extracted. The video clips were accompanied by commentaries that both appear as subtitles on the screen as well as audio produced by an automated text to speech system.[2] The videos are shown in random counter-balanced order to ensure no consistent bias toward segments being shown earlier or later. We asked the judges to score the commentaries using the following metrics:

Score	*Fluency*	*Semantic Correctness*	*Sportscasting Ability*
5	Flawless	Always	Excellent
4	Good	Usually	Good
3	Non-native	Sometimes	Average
2	Disfluent	Rarely	Bad
1	Gibberish	Never	Terrible

Fluency and semantic correctness, or adequacy, are standard metrics in human evaluations of NL translations and generations. Fluency measures how well the commentaries are structured, including syntax and grammar. Semantic correctness indicates whether the commentaries accurately describe what is happening in the game. Finally, sportscasting ability measures the overall quality of the sportscast. This includes whether the sportscasts are interesting and flow well. In addition to these metrics, we also asked them whether they thought the sportscast was composed by a human or a computer (*Human?*).

Since Mechanical Turk recruits judges over the Internet, we had to make sure that the judges were not assigning ratings randomly. Thus, in addition to asking them to rate each video, we also asked them to count the number of goals in each video. Incorrect responses to this question caused their ratings to be discarded. This is to ensure that the judges faithfully watched the entire clip before assigning ratings. After such pruning, there was on average 36 ratings (from 40 original ratings) for each of the 8 videos. Table 6 shows the results. Statistically significant results are shown in boldface.

Results are surprisingly good across all categories with the machine actually scoring higher than the human on average. However, the differences are not statistically significant based on an unpaired t-test ($p > 0.05$). Nevertheless, it is encouraging to see the machine being rated so highly. There is some variance in the human's performance since there were two different commentators. Most notably, compared to the machine, the human's performance on the 2002 final is quite good because his commentary included many details such as the position of the players, the types of passes, and comments about the overall flow of the game. On the other hand, the human's performance on the 2003 final is quite bad because the human commentator

[2]Sample video clips with sound are available on the web at `http://www.cs.utexas.edu/users/ml/clamp/sportscasting/`.

	Commentator	*Fluency*	*Semantic Correctness*	*Sportscasting Ability*	*Human?*
2001 final	Human	3.74	3.59	3.15	20.59%
	Machine	3.89	3.81	**3.61**	40.00%
2002 final	Human	4.13	**4.58**	**4.03**	**42.11%**
	Machine	3.97	3.74	3.29	11.76%
2003 final	Human	3.54	3.73	2.61	13.51%
	Machine	**3.89**	**4.26**	**3.37**	19.30%
2004 final	Human	4.03	4.17	3.54	20.00%
	Machine	4.13	4.38	4.00	**56.25%**
Average	Human	3.86	4.03	3.34	24.31%
	Machine	3.94	4.03	3.48	26.76%

Table 6: Human evaluation of overall sportscasts. Bold numbers indicate statistical significance.

was very "mechanical" and used the same sentence pattern repeatedly. The machine performance was more even throughout although sometimes it gets lucky. For example, the machine serendipitously said "This is the beginning of an exciting match." near the start of the 2004 final clip simply because this statement was incorrectly learned to correspond to an extracted MR that is actually unrelated.

We also elicited comments from the human judges to get a more qualitative evaluation. Overall, the judges thought the generated commentaries were good and accurately described the actions on the field. Picking from the top 5 generated sentences also added variability to the machine-generated sportscasts that improved the results compared with earlier experiments (Chen & Mooney, 2008). However, the machine still sometimes misses significant plays such as scoring or corner kicks. This is because these plays happen much less frequently and often coincide with many other events (e.g. shooting the ball and kickoffs co-occur with scoring). Thus, the machine has a harder time learning about these infrequent events. Another issue concerns our representation. Many people complain about long gaps in the sportscasts or lack of details. Our event detector only concentrates on ball possession and not on positions or elapsed time. Thus, a player holding onto a ball or dribbling for a long time does not produce any events detected by our simulated perceptual system. Also, a short pass in the backfield is treated exactly the same as a long pass across the field to near the goal. Finally, people desired more color commentary (background information, statistics, or analysis of the game) to fill in the voids. This is a somewhat orthogonal issue since our goal was to build a play-by-play commentator that described events that were currently happening.

4 Proposed Research

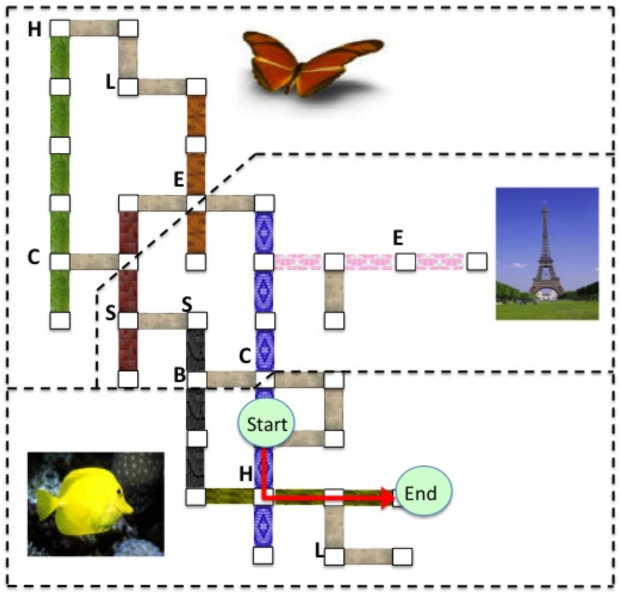
In this section I will describe the proposed work for my thesis, some possible extensions, as well as a long-term view of my research agenda. The proposed thesis work is centered around solving the language grounding problem for a new task: navigating in a virtual environment. Initially I will build a system that learns by passively observing one human giving navigation instructions to another human. I will then explore how to extend the system to learn interactively by playing the role of the instructor or the follower. The long-term goals includes moving from simulations and virtual worlds to dealing with real perceptions such as photos or videos. They also include adapting the techniques we have developed to other NLP tasks such as machine translation.

4.1 Learning Navigation Instructions

While our sportscasting system was able to solve the problem of ambiguous supervision fairly well, the domain was restricted and the ambiguity level was low. Thus, we will extend our approach to a more challenging task: learning navigation instructions in a virtual environment. The navigation task has long been of interest to the linguistics community (Anderson, Bader, Bard, Boyle, Doherty, Garrod, Isard, Kowtko, McAllister, Miller, Sotillo, Thompson, & Weinert, 1991). Recently, the Generating Instructions in Virtual Environments (GIVE) challenge was held that judged how well human followers could navigate and retrieve a trophy in a video game environment using instructions generated by computers (Byron, Koller, Striegnitz, Cassell, Dale, Moore, & Oberlander, 2009). There has also been some work in mapping natural language instructions into actions that can be executed by a computer or a robot (MacMahon, Stankiewicz, & Kuipers, 2006; Branavan et al., 2009; Lau et al., 2009). However, all of these works rely on hand-built systems or token similarities between the instruction and the action parameters. In contrast, our goal is to build a language-independent system that learns how to semantically parse and generate navigation instructions. In particular, we build on the work of MacMahon et al. to construct our task.

The task as described by MacMahon et al. (2006) consists of two participants, an instructor and a follower. Using predefined locations, the instructor gives a set of written instructions for how to navigate from one location to another. The follower then reads the instructions and tries to navigate to the indicated location without any further interactions with the instructor. An example of such instruction is shown in Figure 10. There are three virtual worlds total, each consisting of many interconnecting hallways with different floor tiles and wallpapers. There are also objects placed through out the worlds that could be used as landmarks. MacMahon et al. manually constructed a system called MARCO that is able to parse the NL instructions into a logical language (an example is shown in Figure 10) and then execute the actions in the virtual environment. Evaluating on a large number of navigation instructions given by many different instructors indicated MARCO was able to follow navigation instructions fairly well. We will retain the action execution part of MARCO and replace the parsing component with a learning system.

The training data for the navigation task includes paired human instructions and observed actions (walk forward, turn left, turn right) of the instruction followers. Notice these actions are usually on a lower level than the actions people describe (e.g. *follow this hallway*, *take the first left*, etc). We distinguish between these two types of actions by explicitly referring to the first one as *primitive actions* and the second type as simply actions or *composite actions* since they are composed of several primitive actions. While the formal language designed by MacMahon et al. could be use as our MRL, it was designed to more closely resemble natural language which allows arbitrary nesting of structures as well as several equivalent ways of expressing the same action. This makes it quite difficult to learn as the data is sparse. Thus, we will use only a subset of their formal language as our MRL. Regardless of the representation, the goal of the task is



(a) Map of the world (not seen by participants)

```
When looking away from the wall, go right
and take your 1st left.

Go all the way down until you hit a dead end.


Turn(direction=[Right], precond=Face
(faced=Verify(desc=[Thing(dist='0',
value=Wall, type='Obj', side=[Back])])))

Turn(direction=[Left], location=Travel
(until=Verify(desc=[Thing(dist='0', Part=
[Thing(Path_n=Path, dist='0', type='Path',
side=[Left], value=Path)],
value=Intersection, Order_adj=[1],
type='Struct', side=[At])])))

Travel(distance=[Distance()], until=Verify
(desc=[Thing(dist='0', value=DeadEnd,
type='Struct', side=[At])]))
```

(b) Sample instruction and the corresponding semantic form

Figure 10: This is an example of a navigation instruction in our virtual environment. The world consists of interconnecting hallways with varying flooring and wallpapers (butterfly, fish, or Eiffel Tower.) Letters indicate objects (e.g. 'C' is a chair) at a location.

to learn how to semantically parse natural language instruction into MRs so the robot can execute them to reach the intended destination. Additionally, the system should learn how to generate a set of instructions to guide a human to a particular location. We will evaluate our end system on both of these tasks.

The main challenge with learning navigation instructions is that the level of ambiguity is a lot higher than for the sportscasting task. When asked to give instructions for the same route, different instructors will give the instructions differently. Below are some examples of instructions given by different instructors for the same route as shown in Figure 10. The starting and ending locations are referred to as position 3 and 4, respectively.

> Turn around and look around for a coat hanger. Go towards the coat hanger and turn left at it. Go straight down the hallway and the dead end is position 4.
>
> With your back to the wall at 3 you see an alley in front and blue carpet to each side. Turn right and walk to the hat rack. Turn left. The carpet should have green octagons. Go to the end of this alley. This is p-4.
>
> Turn so that the wall is on your right side. Walk forward once. Turn left. Walk forward twice.

As can be seen from the examples, there are many different ways to describe the same simple route. The instructors can segment the action sequence differently by combining a few primitive actions into one instruction. They may choose to use different *parameters* to describe the same action (e.g. *walk to the hat rack* vs. *walk forward once* vs. *take your first left*). They may also use different words and phrases to describe the same parameters (e.g. *coat hanger* vs *hat rack*). All these potential differences multiply together to form an intractable space to exhaustively search as we have done in the sportscasting domain.

Another difficulty of this task is that the MRs are no longer directly observed. In the navigation domain, the MRs represent plans for moving from one location to another. Unlike the sportscasting domain where we were able to perceive and extract the events that the commentator was talking about, these plans are never observed from the instructor or the follower. Instead, we can only see the primitive actions the followers perform while carrying out the plans. Thus, we must infer the MRs from these primitive action sequences.

While this task is harder than the sportscasting task with respect to the ambiguity level and the unobservable MRs, the ability to execute the MRs provides feedback which was unavailable before. In the sportscasting domain, we simply picked the most likely MR out of a possible set based on our models without any external validation. However, the navigation task allows us to act out the MRs and observe the outcome. We could then evaluate whether the outcome was what we expected based on the observed actions of the human instruction followers.

An additional advantage of the navigation task is that it allows for interactive learning. Instead of just passively observing human followers following human instructions, our system could also engage in the interactions directly by playing either the role of the instructor or the follower. While the original task described by MacMahon et al. (2006) does not allow for interactions between the instructor and the follower beyond the written instructions, we could easily set up the environment to do so. For example, the instructor could monitor the progress of the follower and either give simple feedback about whether the follower is on the right path or even give completely new instructions based on the current location of the follower. Under this new setting, we devise three different types of learning for our system. Initially, the system does not have any knowledge of the language so cannot be expected to participate. Thus, it will engage in *passive learning* by observing human interactions. After it has gained sufficient proficiency in the language, it can then participate in *interactive learning as the instructor* or *interactive learning as the follower*. While it is the instructor, it will generate instructions to guide the human follower to an intended location. When the human follower does not act as expected, this provides negative feedback for the instructions that were generated. The system should then update accordingly to avoid future mistakes. When the system acts as the follower, it will try to parse and execute the human instructions. Whenever the human instructor explicitly indicates that it is on the wrong path or implicitly implies so by giving new instructions, the system should treat the previous parse as a negative example.

We will approach solving the navigation task as before, by simultaneously aligning NL instructions to MRs and building translation models between the two. Much like how WASPER concentrated on the parsing task and WASPER-GEN concentrated on the generation task, we propose two different algorithms for solving the problem, one focusing on the instruction parsing process, and the other on the instruction generation process. Despite their different focuses, they both establish NL–MR pairs which can be used as supervised data to train semantic parsers and tactical language generators. The two algorithms can also be combined in various ways to jointly solve the problem. However, since the training data is quite noisy these algorithms might have difficulties finding the correct direction initially. Thus, we also consider an optional word learning component that will reduce the amount of ambiguity.

4.1.1 Modeling the Instruction Parsing Process

In this section I will discuss an algorithm that models the instruction parsing process. First, we will note how humans perform this process on a high level. We start out with a set of instructions written by the human instructors. Each of the human followers reads the instructions and parses them into a general plan of how to navigate. The follower then executes each step of the plan sequentially in the virtual environment and adjusts their actions based on what is actually perceived from the environment. Any ambiguities or mistakes in the original instructions or that arose from the parsing process may be corrected during this stage. For example, if the instruction says *walk forward two steps and make a right* and the opening on the right is actually three steps aways, the follower can correct the mistake in the instruction and infer that the instructor really meant walk three steps.

We will model this entire process in two stages: semantic parsing and navigation. The system will first try to parse the natural language instructions into the most likely MR. The action execution component of MARCO (MacMahon et al., 2006) will then be used to execute the MR in the virtual environment. This produces an observed sequence of actions. Our goal then is to train the system such that this sequence of actions is as close as possible to those observed from the human followers. We will assume the navigation task is fairly well solved and concentrate on the first task which is semantic parsing.

The semantic parsing component can be thought of as providing proposals for possible MRs. The navigation component then acts as a filter to select the MRs that result in the best action sequences. Thus, the semantic parser should be biased to have high recall since the list of MRs it provides will be refined. Psuedocode for the overall algorithm is shown in Algorithm 5. From our experience on the sportscasting task, we know that in order to build a good semantic parser, we will need to establish good examples of NL–MR pairs. However, unlike the sportscasting task, the MRs are completely unobserved in the training data.

Since the MRs are never observed, we have to infer them from the observed primitive action sequences. However, the space of possible MRs that can be inferred is quite large. Thus, we will prune that space down to a reasonable set using our experience from participating in the GIVE challenge (Byron et al., 2009). We first group all contiguous primitive actions of the same type together (e.g. several walk forward actions should be grouped together) to form a sequence of composite actions. We then define the set of possible parameters for each composite action. While our MRL grammar gives us a set of legal parameters, we still have to prune that space down to a list of parameter types and values that actually work. Initially, we could simulate perception to capture all the landmarks that could be seen during the course of an action. This provides us with a reasonable list of parameter values. We could further specify rules that govern what are the possible values for each parameter type. For example, the precondition of an action should only include landmarks that could be observed at the beginning of the action, and the postcondition should include landmarks that could be observed at the end of the action. Other parameters such as the number of steps or the direction to turn can also be specified quite easily.

To build an initial semantic parser, we will need to estimate potential MRs for each NL instruction (lines 3-7). Since we prefer high-recall parsers, we will pair each instruction with the *most specific MR* which is a compact representation of all the possible MRs we consider (lines 29-35). As described in the last paragraph, the most specific MR represents a sequence of composite actions that groups primitive actions of the same type together. Each of the composite action is described using all the possible parameters we allow. This MR is the most specific because it includes all the possible information that could be used to specify the navigation plan. Below is an example of the most specific MR for the first travel action for the example shown Figure 10.

> Travel(Precondition=(Right=Wall, Left=Concrete Hall, Front=Blue Hall, Back=Blue Hall), Distance=1, Until=(Hat Rack, Intersection(Order=1, Current Path=Blue Hallway, Cross Path=Yellow Hallway),), Postcondition=(Right=Yellow Hall, Left=Yellow Hall,Front=Blue Hall, Back=Blue Hall))

After we obtain an initial semantic parser, we can iteratively alternate between training semantic parsers and producing NL–MR pairs (lines 9-26). We first use the semantic parser to produce a set of possible MRs for each instruction. We can then try to execute each of these MRs and observe which produces the same

Algorithm 5 AN ALGORITHM THAT MODELS THE INSTRUCTION PARSING PROCESS

input Pairs of instructions and observed sequences of primitive actions
output *BestExamplesSet*, a set of NL-MR pairs,
TranslationModel, a WASP translation model

```
1:
2: main
3:    //Initial training loop
4:    for pair $(instr, pActions)$ in the training data do
5:       Add $(instr, MostSpecificMR(pActions)$ to $InitialTrainingSet$
6:    end for
7:    $TranslationModel \leftarrow$ Train WASP on $InitialTrainingSet$
8:
9:    //Iterative retraining
10:   repeat
11:      for pair $(instr, pActions)$ in the training data do
12:         $TopMRs \leftarrow$ The top parses of $instr$ according to $TranslationModel$
13:         for meaning representation $m$ in $TopMRs$ do
14:            if Executing $m$ results in $pActions$ then
15:               add $(instr, m)$ to $PotentialMRs$
16:            else
17:               Remove all parameters in $m$ that does not appear in $MostSpecificMR(pActions)$
18:               Add to $PotentialMRs$ the set of MRs obtained by systematically adding parameters in
                  $MostSpecificMR(pActions)$ to $m$ and when executed results in $pActions$
19:            end if
20:         end for
21:         Rank the MRs in $PotentialMRs$ first by parsing score, then by the number of parameters added,
            then by shortest length
22:         $m \leftarrow$ The best MR in $PotentialMRs$
23:         Add $(instr, m)$ to $BestExamplesSet$
24:      end for
25:      $TranslationModel \leftarrow$ Train WASP on $BestExamplesSet$
26:   until Convergence or MAX_ITER reached
27: end main
28:
29: function MostSpecificMR($pActions$)
30:    for Contiguous group of primitive actions $a_i \in pActions$ do
31:       $param_i \leftarrow$ The set of parameters that could be used to describe $a_i$
32:       $m_i \leftarrow$ An MR with action $a_i$ and parameters $param_i$
33:    end for
34:    return An MR constructed by concatenating all $m_i$
35: end function
```

action sequence as in the training data. The top MR that produces the correct action is then paired with the instruction as an example NL–MR pair that we can use to train a new semantic parser.

However, given that the space of possible MRs is quite large and the quality of the initial semantic parser is expected to be low, it is possible that none of the top ranked parses produced by the semantic parser will produce the correct actions. Thus, we will also try to improve the top MRs by adding or removing constraints (lines 16-19). First, we will remove any constraints not included in the most specific MR. Then we will systematically add constraints in the most specific MR until the correct actions are produced. In the case that multiple MRs produce the desired action sequences, we will first prefer the MRs with the least modifications and then the shortest MR. The first bias is to help the semantic parser converge. While the modifications will eventually lead to the correct action sequence, they may produce a MR that is completely different than what was described in the instructions. Thus, we prefer to use a MR that is closer to what the semantic parser produced. The second bias is to ensure that we do not produce any superfluous constraints that were not necessary in performing the actions. Of course, it is possible that the instructions had actually specified redundant constraints for robustness, but it is better to learn to parse only part of the instructions that is sufficient rather than to introduce incorrect associations.

4.1.2 Modeling the Instruction Generation Process

While we have described how to model the instruction parsing process, another goal of ours is to learn to generate navigation instructions. Thus, I will discuss a different algorithm that models the instruction generation process in this section. Again, we will note how humans perform this task on a high level. Initially, the instructor navigates around the environment, building a mental map of all the landmarks and locations. When asked to give instructions to get from one location to another, the instructor first forms the action sequence required to get to the destination. Note these actions can be at much higher levels than the primitive actions (e.g. *follow this path* or *find the blue corridor*.) Then the instructor selects a subset of the possible parameters to describe each of the actions. Finally, each action along with its parameters is formulated into a natural language instruction in the context of the preceding instructions.

We will model this process in two stages as well: strategic generation and tactical generation. The system first needs to select an action to verbalize. Then for each action, the system has to choose a set of parameters to describe it. In other words, the system needs to decide what to talk about. The content of the description will be formalized using our MRL. Finally, we will use a tactical language generator to transform the MR into a natural language instruction. There are several objectives we could try to optimize here. The first is the standard maximization of the likelihood of the data. In other words, we want the system to produce the given navigation instructions with high probability. However, given the variability of language, it may be more suitable to maximize the generation score (BLEU/NIST/METEOR) of the system's output when compared to the original instructions. In this manner, our objective becomes to make the most likely generation to be as close to the human instructions as possible. To make the process even more instructor-independent, we could instead aim for task completions. In other words, we could ask a follower to follow the produced instructions and observe whether the follower reaches the intended destination. Consequently, the produced outputs do not need to look anything like the original instructions. Initially, we can use the full MARCO system to automate this evaluation process. Specifically, the MARCO system will parse our generated instructions and execute the parsed actions. Eventually we could use an Internet-based approach to recruit human subjects to participate as followers. One drawback to the purely task-based evaluation is that it may result in correct but aesthetically unpleasant, fragile, or inefficient instructions. For example, a pure step-by-step instruction of the required actions (e.g. *go forward two steps, turn right, go forward three steps, turn left, go forward one step, turn right, etc*) is sufficient to get the follower to the intended destination. However, it is hard for the follower to remember and can be very verbose since each action needs to be specified. Thus, ideally we want the system to produce instructions that accomplish the task but also resemble the training data.

A graphical representation of the entire instruction generation model can be seen in Figure 11. Since the actual action sequence is not directly observed because it only exists in the mind of the instructor, we will use the observed sequence of primitive actions. The instructions are also observed. Thus, we are left to infer the actions and the associated parameters that were chosen to generate the instructions.

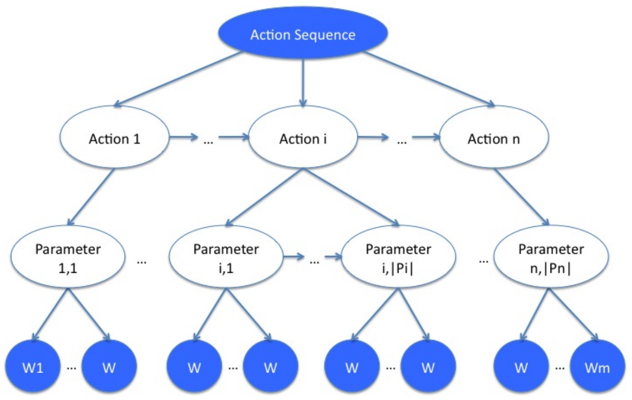


Figure 11: Generative model of how instructions are produced. First, actions are selected and ordered from the entire action sequence. Then for each action, a list of parameters are chosen to describe the action. Finally, words are chosen for each parameter. The action sequence and the words are given while the actions and the parameters have to be inferred.

The strategic generation component encompasses the first two levels of the generative model. Given the action sequence, we want to define a distribution over all ordered subsequences of actions and all ordered sequence of parameters. The tactical generation component is the last stage of the generative model where words are generated given the parameters. We will now discuss each stage of the model in more detail.

- **Action Choice Model** The action model is intended to capture our intuition about which actions are usually used in navigation instructions. Similar to how we derived the most specific MR in the previous section, we will constrain the valid action sequences allowed by our model given the observed primitive action sequence $\mathbf{p}$. We first group all contiguous primitive actions of the same type together to form the composite action sequence $\mathbf{c} = (c_1, ..., c_{|\mathbf{c}|})$. We then define a valid action sequence to be a segmentation of $\mathbf{c}$ where each segment has size 0, 1, or 2. A segment of size zero maps to a special action called $Verify$ which does not correspond to any primitive actions and is used to verify certain conditions in the environment (e.g. *You should see a chair to your left*). Thus, the set of possible actions is then $\{Verify, Forward, Left, Right, Forward+Left, Forward+Right, Left+Forward, Left+Right, Right+Forward, Right+Left\}$. We then define a probability function for all valid action sequences $\mathbf{a} = (a_1, ..., a_n)$ formally as follows:

$$P(\mathbf{a}|\mathbf{p}) = \prod_{i=1}^{n} P(a_i|a_{i-1})$$

For each action node, it can take one of three values. It can be $Verify$, the next action, or the next two actions. The probability of each value being chosen is only dependent on the value of the last action node.

- **Parameter Choice Model** After we have selected an action, we need to choose the set of parameters to describe this action. This can include the preconditions that need to be met before executing the action, parameters describing the details of the action (e.g. number of steps to take or the direction to turn), or postconditions that specify when the action should stop. The parameter choice model is designed to capture the prior probability of a particular parameter being used to describe an action as well as the usual order of the parameters. For example, for the action of going forward, typically the instructor will specify where to stop or the number of steps to take and only occasionally mention the objects the follower will pass on the way. Moreover, if both the stopping condition and the number of steps to take are present, usually the latter will appear first (e.g *Go forward three steps to where the couch is* and not *Go forward to where the couch is after three steps*). This is consistent with the observation that people describe things in the order they are processed. It is more natural to count the number of steps as the person walks and then check that they have arrived at the correct location by verifying that the stopping condition is met rather than the other way around.

  Once again, we will limit the possible values of parameters to ones that can actually lead to the correct primitive actions. This process will be done similar to how we created the most specific MR in the previous section. We will specify rules for constructing valid parameter values based on the landmarks that can been seen during the course of the action. Other parameters such as the number of steps or the direction to turn can also be specified quite easily.

  We will now formally define the distribution over all possible sequences of parameters $\mathbf{p} = (p_1, ..., p_m)$ where each $p_i \in validParam(a)$. The function $validParam(a)$ specifies all the valid parameters for the action a. The distribution is then

$$P(\mathbf{p}|a)) = \prod_{i=1}^{m} P(p_i|p_{i-1})$$

- **Word Choice Model** Finally, we have to generate words given the parameters of an action. We will use parameter-specific language models with smoothing to generate the word sequences associated with each parameter. To deal with data sparsity, we will use a back-off model both at the language model level and at the global level. In other words, we will first try to use a language model specific to the parameter type and parameter value. Then we will gradually back-off the specific parameter value to partial matches of the value, then to just using the parameter type, and finally using just the action type.

- **Tactical Generation Model** An alternative to the word choice model we have just described is to use a tactical generation model to generate the words. Given that we have already generated all the actions and their associated parameters, we could define a generative model that translates the MR into NL instructions. In this way, we will produce the instructions jointly from all action and all parameters instead of having each parameter producing words independently. However, this model may be more difficult to build initially since it will require more data to estimate good parameters.

To train the generative model, we will use the EM algorithm which alternatively calculates expected counts of the inferred actions and parameters and optimizes the parameters of each of the stages given the inferred counts. We will initialize the model with uniform distributions over all the choices.

4.1.3 Combining the Models

While we have described two independent models of the training data, we could also mix the components together. Instead of each model estimating the hidden MRs on its own, they could perform the estimation jointly. The semantic parsing and the strategic generation components can be used to propose potential MRs while the navigation and the tactical generation components can be used to verify the quality of the proposals. Given an instruction and the corresponding observed actions, the semantic parser and the strategic generator could each produce a set of potential MRs. The navigation component could then verify whether executing these MRs result in the observed actions. On the other hand, we could also evaluate the quality of the MRs using the tactical generator by calculating the generation scores or the probability of generating the given instructions. Finally, we could either combine the scores from all four components by multiplying them together or use a voting scheme to select the best MRs. Then each component could retrain on these MRs weighted by their scores.

Alternatively, we can also employ a co-training approach (Blum & Mitchell, 1998). Instead of jointly scoring all the MRs and training all the components on the best ones, we can keep the instruction parsing and the instruction generation models separate. The top ranked MRs from each of the algorithms are passed to the other one for training at each iteration. While this does not fit in the traditional framework of co-training which requires two different views of the same data, it is similar in spirit in that we have two independently trained systems both of which are capable of labeling new data. The individual bias of the systems is ideally ameliorated by using the training data produced by the other system.

4.1.4 Word Learning

Given that the space of MRs we are exploring is quite large, it may be difficult to learn how to parse and generate entire MRs directly. Thus, we will also consider first learning a lexicon that helps us reduce the amount of ambiguity. In particular, it will be useful to learn words that correspond to actions and words that correspond to parameters. It is not necessary to learn exact meanings of the words, as long as they provide useful bias toward selecting the correct MRs in our overall model.

One way to learn the meaning of the word is by taking the intersection of all MRs they are associated with (Thompson, 1998). For all the examples the word occurs in, we generate the most specific MR for that example. Then we take the intersection of all such MRs to determine the meaning of the word. However, since the data is noisy, we will also attempt to first induce more structures in the NL instructions. For example, we could cluster the words based on their part of speech or semantic similarity (e.g. appears in similar context). Then we would only need to focus on learning words that are likely to refer to actions or parameters. Moreover, we could exploit the order of the words. As mentioned before, people generally give instructions in the same order they are executed. Thus, an object that appears early in the path would also likely be mentioned early in the instruction. Adding such constraints could lead us to learn the correct lexicon.

Once we have learned a good lexicon, we could then use it to prune down the list of possible MRs we consider for each example. For example, if we know the meaning of the word *sofa*, we could remove all MRs that do not refer to the sofa. Alternatively, we could also remove MRs that mention actions or parameters not covered by any of the words. For the first scenario, we want a high-precision lexicon while for the second scenario, we want a high-recall lexicon.

4.2 Extension to Learning in an Interactive Setting

In addition to passively learning from instructions paired with observed action sequences, the navigation task also allows for an interactive setting of learning. Once we have built a system that could perform the semantic parsing and language generation tasks decently, we could allow the system to participate in the task directly by playing either the role of the instructor or the follower and let the human participant play the other role. The feedback from the human partner could then be used to construct positive and negative training examples for improving the system.

In the case that the system is the instructor, feedback is natural and automatic. Any time the human follower deviates from the intended action sequence, negative feedback is received. When the follower successfully completes an instruction, positive feedback is received. Thus, the human follower does not have to explicitly provide any feedback about the quality of the instructions.

On the other hand, when the system is the follower, the human instructor needs to be more explicit in giving negative feedback. While the system can infer that it has done something incorrectly whenever the instructor gives updated new instructions, such cue may be noisy and unreliable. Thus, it is more likely the instructor will have to manually indicate when the robot is doing the wrong thing. Positive feedback, however, is still automatically received when the robot completes an instruction without any negative feedback.

A simple way to utilize the feedback is to retain the positive examples as training data. The system can then periodically retrain itself using the collected positive examples. However, this only reinforces solving instances that the system already solved correctly. Depending on the learning algorithm, we could potentially also train on the negative examples. For example, KRISP uses a SVM classifier so the negative examples can directly become part of the training data. Each of the production rules that were used to create the bad parse will treat the substring they covered as a negative example. On the other hand, it is unclear how to incorporate negative examples into the learning process of WASP. Building SCFG rules with negative weights simply means those rules will be ignored. Nevertheless, we can also force the system to find the correct solution for all occurrences of negative feedback. So in the case that the system is generating instructions, it can try generating different instructions until the follower performs the correct action. Similarly, the system can try different actions when it is the follower until the human instructor is satisfied.

So far we have assumed the way to influence the system is by establishing training examples of NL–MR pairs and re-train the system at each iteration. However, it may be desirable to be able to update the system in an online manner without training from scratch. The system can then adapt more quickly based on the feedback it receives. For example, negative feedback can be used to change WASP's SCFG rules directly by lowering the weights of the rules that were used that led to the negative feedback. In this manner, it is no longer necessary for the system to attempt to solve a failed instance repeatedly until it finds the correct solution.

4.3 Long-term Future Directions

This section describes some of the long-term goals of my research that extends beyond the scope of my thesis. They span across several different directions including extensions to the already described language grounding systems, working with real perception data, and applying the technique developed for language grounding to other NLP tasks with ambiguous supervision.

4.3.1 Recognizing Patterns

In the sportscasting problem, some statements in the commentaries specifically refer to a pattern of activity across several recent events rather than to a single event. For example, in one of the English commentaries, the statement "Purple team is very sloppy today." appears after a series of turn-overs to the other team. The simulated perception could be extended to extract patterns of activity such as "sloppiness;" however this assumes that such concepts are predefined, and extracting many such higher-level predicates would greatly increase ambiguity in the training data. The current system assumes it already has concepts for the words it needs to learn and can perceive these concepts and represent them in MRs. However, it would be interesting to include a more "Whorfian" style of language learning (Whorf, 1964) in which an unknown word such as "sloppiness" could actually cause the creation of a new concept. For content words that do not seem to consistently correlate with any perceived event, the system could collect examples of recent activity where the word is used and try to learn a new higher-level concept that captures a regularity in these situations. For example, given examples of situations referred to as "sloppy," an inductive logic programming system (Lavrač & Džeroski, 1994) should be able to detect the pattern of several recent turnovers.

The ability to automatically construct higher level abstractions will free us from building a representation that encompasses all possible things we want to talk about. Instead of anticipating all potential actions and events, we could keep the representation at a low level and let the system build new concepts as needed. In particular, in certain continuous domains, we would not have to build specific bins for possible ranges of value. Instead, the system can infer how to categorize these ranges automatically. For example, in the weather report domain, the system can learn the correspondences between temperature ranges and descriptions such as *hot*, *warm*, *cool*, or *cold*.

4.3.2 Active Learning

For the navigation domain, a natural extension to the interactive learning discussed previously is to use active learning (Cohn, Atlas, & Ladner, 1994). Since we work in a virtual environment, we can set up the world any way we want. Consequently, we can manipulate the world in such a way as to allow the system to get feedback about actions or landmarks it is the least confident about. For example, if the system is not sure how to describe a chair, it could purposely place a chair in the environment and use it as a landmark when generating the instruction. The question then becomes how do we choose what actions or landmarks we want to learn about and how do we construct examples that gives us the maximum amount of information. However, we may not always have the ability to change the environment as we desire. Nevertheless, we could still define different tasks in a fixed environment in order to learn.

One possible way to determine what to learn is to sample the space of MRs randomly and generate an instruction for each of the MRs. The MR that produced the least confident output is then selected as the next target for learning. We could then isolate each part of the MR and construct examples that requires understanding that specific part. For example, for any landmarks, we could place them along a long hallway and ask the human follower to walk forward until they reach the landmark. In a fixed environment, we could define a task that requires the follower to find the landmark.

4.3.3 Real Perceptions

An obvious extension to the sportscasting task is to apply it to *real* RoboCup games rather than simulated ones. Recent work by Rozinat *et al.* (2008) analyzes games in the RoboCup Small Size League using video from the overhead camera. By using the symbolic event trace extracted by such a real perceptual system, our methods could be applied to real-world games. Using speech recognition to accept spoken language input is another obvious extension.

For the navigation domain, we could use a real robot to navigate in a real environment. We would need to use an object recognition system to help us identify landmarks in the environment. Once the world has been processed into various tokens of landmarks, we could then use our system to ground the natural language. Here the ability to recognize patterns and automatically build higher level representations can become even more useful. Instead of pre-defining what tokens to learn and use, we can directly process the low level vector representation (e.g. SIFT descriptors of interest points (Lowe, 1999)) and automatically discover objects and the words that are used to describe that object.

4.3.4 Machine Translation

The idea behind WASP is to borrow from statistical machine translation techniques. Thus, it is only natural that we can apply the lessons we have learned back to doing the task of machine translation. We can use our algorithm to build translation models from ambiguously paired parallel corpora. This kind of data can occur when the translation is loose and not sentence-aligned. For example, wikipedia articles often are not aligned for different languages. However, they generally contain much of the same information. Thus, we can build a system that simultaneously solves the alignment problem and builds a translation model. For example, we can first train on sentences in one language matched to every sentence in the corresponding paragraph in the other language. Then we can use the translation model to help decide which sentences are the best matches. We can then iterate this process until convergence. Since the ambiguity level can be high and many sentences may not have matches, we can exploit token similarities or known word translations (e.g. the name of the wikipedia article) to reduce the number of possible alignments.

Another way to use our idea for machine translation is to use the MRL as an intermediate step for translating to any language. In other words, we would first parse the language we want to translate from into a MR. Then, we would use the MR to generate sentences in the language we want to translate to. For translating between n languages, we only need $2n$ models instead of $\frac{n\cdot(n-1)}{2}$ models. Furthermore, we no longer need parallel corpora between the different languages. It is only necessary that they align to the same MRL. In other words, we are aligning the different languages using the world they describe instead of directly to each other. This can be a very useful application for the sportscasting domain where simultaneous broadcasts in many different languages are often required for large international sports events such as the Olympics or the World Cup.

5 Conclusion

Learning the semantics of language from the perceptual context in which it is used is a useful approach because only minimal human supervision is required. Moreover, it is language independent so no changes to the system are required for learning a new language. We demonstrated the utility of such approach by presenting a system that learns to generate sportscasts from only observing sample commentaries and no other human supervisions.

Dealing with the ambiguous supervision inherent in the training environment is a critical issue in learning language from perceptual context. We have evaluated various methods for disambiguating the training data in order to learn semantic parsers and language generators. Using a generation evaluation metric as the criterion for selecting the best NL–MR pairs produced better results than using semantic parsing scores when the initial training data is very noisy. Our system also learns a simple model of strategic generation from the ambiguous training data by estimating the probability that each event type evokes human commentary. Finally, experimental evaluation verified that the overall system learns to accurately parse and generate comments and to generate sportscasts that are competitive with those produced by humans.

The proposed future work is to extend the system to solve the more complex navigation task. Without being able to observe the navigation plans referred to by the instructions, the system has to learn to deal with an exponential number of possible alignments between the NL and the MRs. Combining two separate models of the data, we plan to use the constraints imposed by each of them jointly to reduce the alignment space. The interactive nature of the task also allows us to improve the system by receiving feedback from a human partner about the quality of the system's understanding of the language.

Acknowledgments

We thank Joohyun Kim for helping to run some of the experiments, and Adam Bossy for his work on simulating perception for the RoboCup games. We also thank Percy Liang for sharing his experimental results with us. This work was funded by the NSF grant IIS–0712907X. Most of the experiments were run on the Mastodon Cluster, provided by NSF Grant EIA-0303609.

References

Aho, A. V., & Ullman, J. D. (1972). *The Theory of Parsing, Translation, and Compiling*. Prentice Hall, Englewood Cliffs, NJ.

Anderson, A., Bader, M., Bard, E., Boyle, E., Doherty, G. M., Garrod, S., Isard, S., Kowtko, J., McAllister, J., Miller, J., Sotillo, C., Thompson, H. S., & Weinert, R. (1991). The HCRC map task corpus. *Language and Speech*, *34*, 351–366.

André, E., Binsted, K., Tanaka-Ishii, K., Luke, S., Herzog, G., & Rist, T. (2000). Three RoboCup simulation league commentator systems. *AI Magazine*, *21*(1), 57–66.

Bailey, D., Feldman, J., Narayanan, S., & Lakoff, G. (1997). Modeling embodied lexical development. In *Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society*.

Barnard, K., Duygulu, P., Forsyth, D., de Freitas, N., Blei, D. M., & Jordan, M. I. (2003). Matching words and pictures. *Journal of Machine Learning Research*, *3*, 1107–1135.

Barzilay, R., & Lapata, M. (2005). Collective content selection for concept-to-text generation. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP-05)*.

Berg, T. L., Berg, A. C., Edwards, J., & Forsyth, D. A. (2004). Who's in the picture. In *Advances in Neural Information Processing Systems 17 (NIPS 2004)*.

Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pp. 92–100 Madison, WI.

Branavan, S., Chen, H., Zettlemoyer, L. S., & Barzilay, R. (2009). Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processin (ACL-IJCNLP 2009)*.

Brown, P. F., Cocke, J., Della Pietra, S. A., Della Pietra, V. J., Jelinek, F., Lafferty, J. D., Mercer, R. L., & Roossin, P. S. (1990). A statistical approach to machine translation. *Computational Linguistics*, *16*(2), 79–85.

Brown, P. F., Della Pietra, V. J., Della Pietra, S. A., & Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, *19*(2), 263–312.

Bunescu, R. C., & Mooney, R. J. (2005). Subsequence kernels for relation extraction. In Weiss, Y., Schölkopf, B., & Platt, J. (Eds.), *Advances in Neural Information Processing Systems 19 (NIPS 2006)* Vancouver, BC.

Byron, D., Koller, A., Striegnitz, K., Cassell, J., Dale, R., Moore, J., & Oberlander, J. (2009). Report on the first nlg challenge on generating instructions in virtual environments (give). In *Proceedings of the 12th European Workshop on Natural Language Generation (ENLG 2009)* Athens, Greece.

Chen, D. L., & Mooney, R. J. (2008). Learning to sportscast: A test of grounded language acquisition. In *Proceedings of 25th International Conference on Machine Learning (ICML-2008)* Helsinki, Finland.

Chen, M., Foroughi, E., Heintz, F., Kapetanakis, S., Kostiadis, K., Kummeneje, J., Noda, I., Obst, O., Riley, P., Steffens, T., Wang, Y., & Yin, X. (2003). Users manual: RoboCup soccer server manual for soccer server version 7.07 and later.. Available at `http://sourceforge.net/projects/sserver/`.

Chiang, D. (2005). A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL-05)*, pp. 263–270 Ann Arbor, MI.

Cohn, D., Atlas, L., & Ladner, R. (1994). Improving generalization with active learning. *Machine Learning*, *15*(2), 201–221.

Collins, M. (2002). New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pp. 263–270 Philadelphia, PA.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, *39*, 1–38.

Doddington, G. (2002). Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of ARPA Workshop on Human Language Technology*, pp. 128–132 San Diego, CA.

Fleischman, M., & Roy, D. (2007). Situated models of meaning for sports video retrieval. In *Proceedings of Human Language Technologies: The Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT-07)* Rochester, NY.

Gold, K., & Scassellati, B. (2007). A robot that uses existing vocabulary to infer non-visual word meanings from observation. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI-07)*.

Gorniak, P., & Roy, D. (2005). Speaking with your sidekick: Understanding situated speech in computer role playing games. In *Proceedings of the 4th Conference on Artificial Intelligence and Interactive Digital Entertainment* Stanford, CA.

Gupta, S., & Mooney, R. (2009). Using closed captions to train activity recognizers that improve video retrieval. In *Proceedings of the CVPR-09 Workshop on Visual and Contextual Learning from Annotated Images and Videos (VCL)* Miami, FL.

Harnad, S. (1990). The symbol grounding problem. *Physica D*, *42*, 335–346.

Herzog, G., & Wazinski, P. (1994). VIsual TRAnslator: Linking perceptions and natural language descriptions. *Artificial Intelligence Review*, *8*(2/3), 175–187.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the Tenth European Conference on Machine Learning (ECML-98)*, pp. 137–142 Berlin. Springer-Verlag.

Kate, R. J., & Mooney, R. J. (2006). Using string-kernels for learning semantic parsers. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL-06)*, pp. 913–920 Sydney, Australia.

Kate, R. J., & Mooney, R. J. (2007). Learning language semantics from ambiguous supervision. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI-07)*, pp. 895–900 Vancouver, Canada.

Knight, K., & Hatzivassiloglou, V. (1995). Two-level, many-paths generation. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL-95)*, pp. 252–260 Cambridge, MA.

Kruijff, G.-J. M., Zender, H., Jensfelt, P., & Christensen, H. I. (2007). Situated dialogue and spatial organization: What, where... and why?. *International Journal of Advanced Robotic Systems*, *4*(2).

Lau, T., Drews, C., & Nichols, J. (2009). Interpreting written how-to instructions. In *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence (IJCAI-2009)*.

Lavrač, N., & Džeroski, S. (1994). *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood.

Liang, P., Jordan, M. I., & Klein, D. (2009). Learning semantic correspondences with less supervision. In *Proceedings of the Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processin (ACL-IJCNLP 2009)*.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., & Watkins, C. (2002). Text classification using string kernels. *Journal of Machine Learning Research*, *2*, 419–444.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Vol. 2.

Lu, W., Ng, H. T., Lee, W. S., & Zettlemoyer, L. S. (2008). A generative model for parsing natural language to meaning representations. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP-08)* Honolulu, HI.

MacMahon, M., Stankiewicz, B., & Kuipers, B. (2006). Walk the talk: Connecting language, knowledge, and action in route instructions. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06)*.

Mooney, R. J. (2007). Learning for semantic parsing. In Gelbukh, A. (Ed.), *Computational Linguistics and Intelligent Text Processing: Proceedings of the 8th International Conference, CICLing 2007, Mexico City*, pp. 311–324. Springer Verlag, Berlin.

Och, F. J., & Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics*, *29*(1), 19–51.

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pp. 311–318 Philadelphia, PA.

Riezler, S., Prescher, D., Kuhn, J., & Johnson, M. (2000). Lexicalized stochastic modeling of constraint-based grammars using log-linear measures and EM training. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-2000)*, pp. 480–487 Hong Kong.

Roy, D. (2001). Learning visually grounded words and syntax of natural spoken language. *Evolution of Communication*, *4*(1).

Roy, D. (2002). Learning visually grounded words and syntax for a scene description task. *Computer Speech and Language*, *16*(3), 353–385.

Roy, D. (2005). Semiotic schemas: A framework for grounding language in action and perception. *Artificial Intelligence*, *167*, 170–205.

Roy, D., & Pentland, A. (2002). Learning words from sights and sounds: a computational model. *Cognitive Science*, *26*(1), 113–146.

Rozinat, A., Zickler, S., Veloso, M., van der Aalst, W., & McMillen, C. (2008). Analyzing multi-agent activity logs using process mining techniques. In *Proceedings of the 9th International Symposium on Distributed Autonomous Robotic Systems (DARS-08)* Tsukuba, Japan.

Satoh, S., Nakamura, Y., & Kanade, T. (1997). Name-it: Naming and detecting faces in video by the integration of image and natural language processing. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*.

Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Shieber, S. M. (1988). A uniform architecture for parsing and generation. In *Proceedings of the 12th International Conference on Computational Linguistics (COLING-88)*, pp. 614–619 Budapest, Hungary.

Siskind, J. M. (1996). A computational study of cross-situational techniques for learning word-to-meaning mappings. *Cognition*, *61*(1), 39–91.

Snyder, B., & Barzilay, R. (2007). Database-text alignment via structured multilabel classification. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-2007)*.

Srihari, R. K., & Burhans, D. T. (1994). Visual semantics: Extracting visual information from text accompanying pictures. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*.

Stolcke, A. (1995). An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, *21*(2), 165–201.

Thompson, C. A. (1998). *Semantic Lexicon Acquisition for Learning Natural Language Interfaces*. Ph.D. thesis, Department of Computer Sciences, University of Texas, Austin, TX. Also appears as Artificial Intelligence Laboratory Technical Report AI 99-278 (see http://www.cs.utexas.edu/users/ai-lab).

Whorf, B. L. (1964). *Language, Thought, and Reality: Selected Writings*. MIT Press.

Wong, Y., & Mooney, R. J. (2006). Learning for semantic parsing with statistical machine translation. In *Proceedings of Human Language Technology Conference / North American Chapter of the Association for Computational Linguistics Annual Meeting (HLT-NAACL-06)*, pp. 439–446 New York City, NY.

Wong, Y., & Mooney, R. J. (2007). Generation by inverting a semantic parser that uses statistical machine translation. In *Proceedings of Human Language Technologies: The Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT-07)*, pp. 172–179 Rochester, NY.

Yamada, K., & Knight, K. (2001). A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL-2001)*, pp. 523–530 Toulouse, France.

Yu, C., & Ballard, D. H. (2004). On the integration of grounding language and learning objects. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-04)*, pp. 488–493.

Zaragoza, H., & Li, C.-H. (2005). Learning what to talk about in descriptive games. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP-05)*, pp. 291–298 Vancouver, Canada.

Zelenko, D., Aone, C., & Richardella, A. (2003). Kernel methods for relation extraction. *Journal of Machine Learning Research*, *3*, 1083–1106.

Zettlemoyer, L. S., & Collins, M. (2005). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of 21st Conference on Uncertainty in Artificial Intelligence (UAI-2005)* Edinburgh, Scotland.

Zettlemoyer, L. S., & Collins, M. (2007). Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL-07)*, pp. 678–687 Prague, Czech Republic.